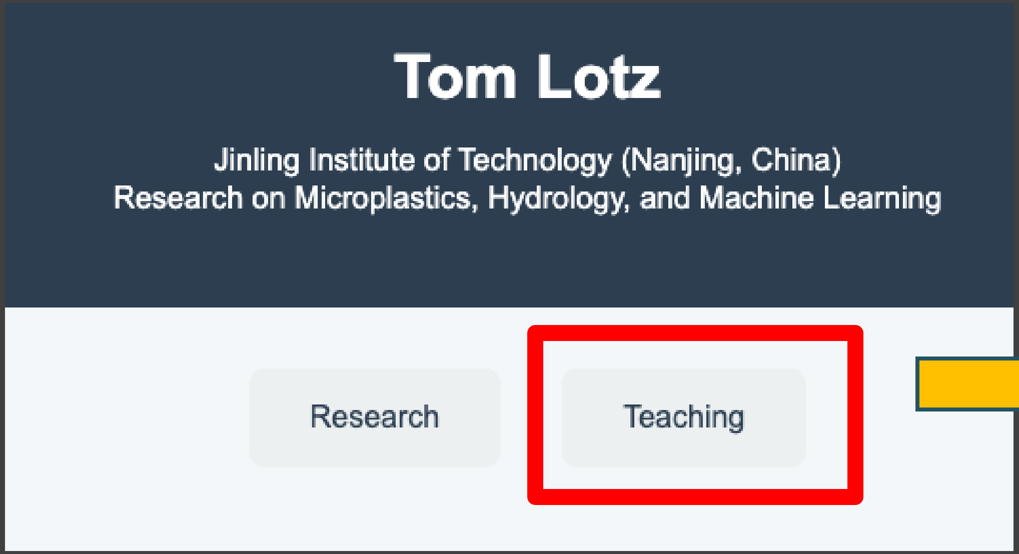
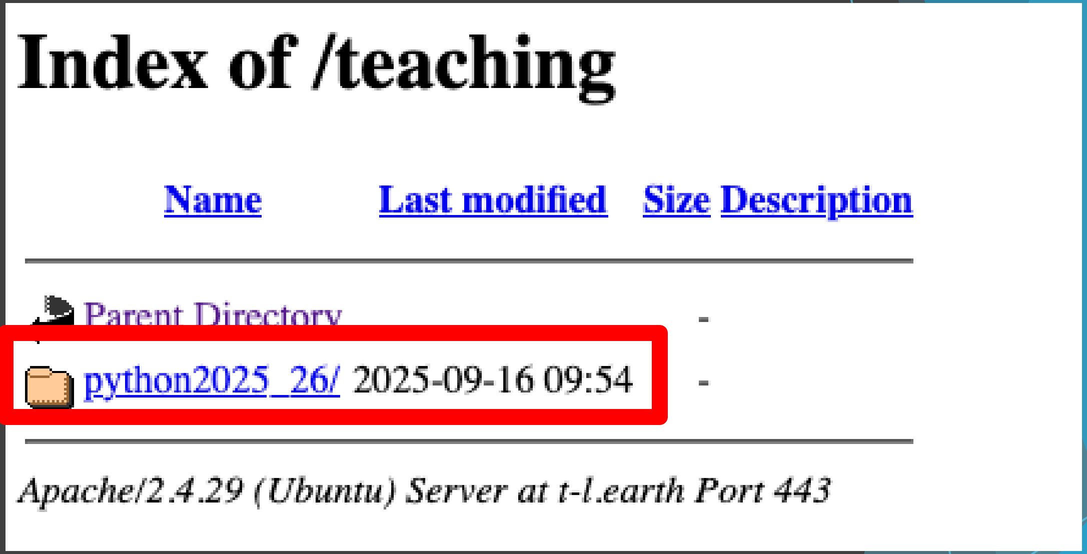
Download the cheat sheets and slides from here

t-l.earth



Tom Lotz

Jinling Institute of Technology (Nanjing, China)
Research on Microplastics, Hydrology, and Machine Learning



Index of /teaching

```
      Name           Last modified      Size Description
Parent Directory                         -
python2025_26/  2025-09-16 09:54         -
```

Apache/2.4.29 (Ubuntu) Server at t-l.earth Port 443

Python语言程序设计

Python Programming

2025/26

Session 05

Tom Lotz (tom.lotz@outlook.com)

Content

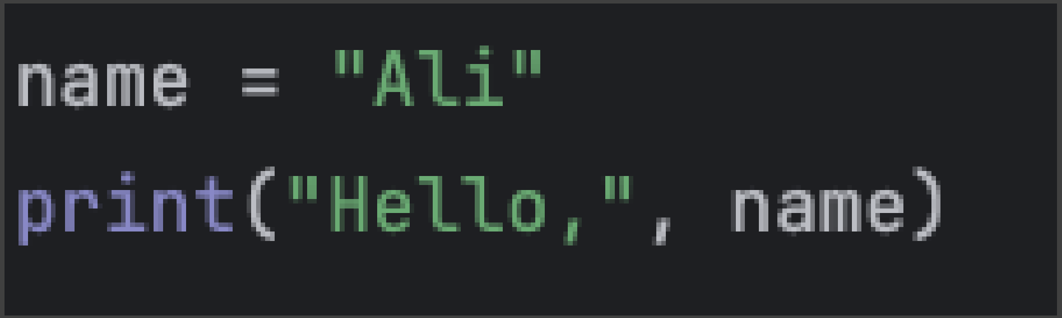
Brain activation

PyCharm

- Start your PyCharm, on your own laptop, or the computer in front of you.
- If none of these work, use www.online-python.com

Hello Again!

- Write this code and run it:



```python
name = "Ali"
print("Hello,", name)
```

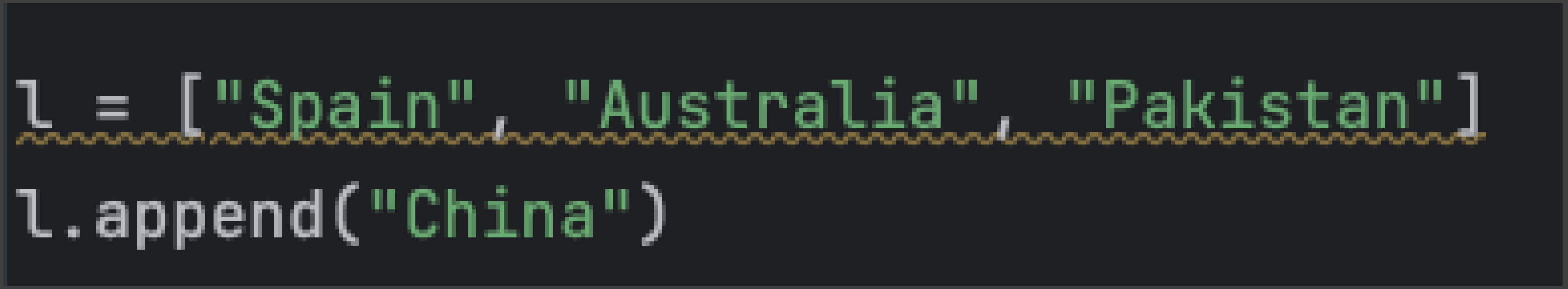
Math

- Create two variables a = 5 and b = 3. Print their sum, difference, and product on separate lines.

Lists

- Make a list of 3 favorite fruits. Print the first and last fruit. Then use .append() to add one more fruit and print again.

- Reminder:

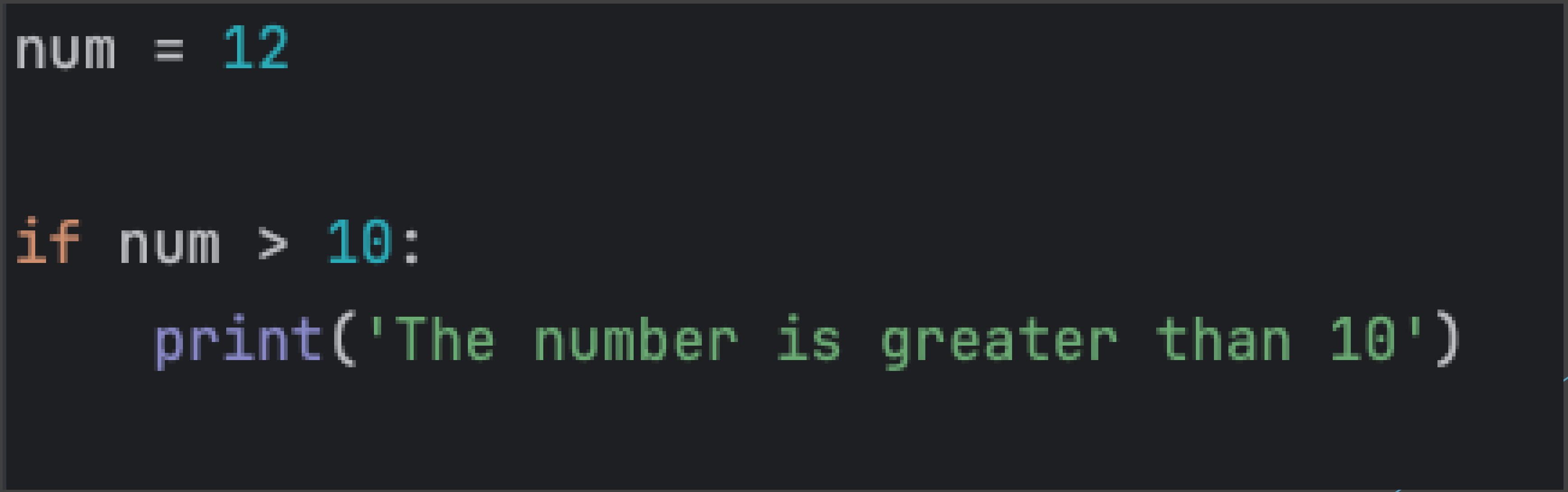


```python
l = ["Spain", "Australia", "Pakistan"]
l.append("China")
```

Conditionals

- Make a variable called num. Assign a number. Then check if the number is negative or positive and print the result.

- Reminder:



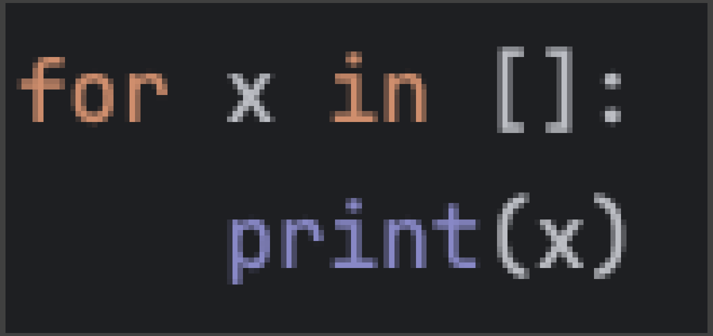
```python
num = 12


if num > 10:
    print('The number is greater than 10')
```

Loops

- Make a list of numbers and then use a for loop to print all of them.

- Reminder:

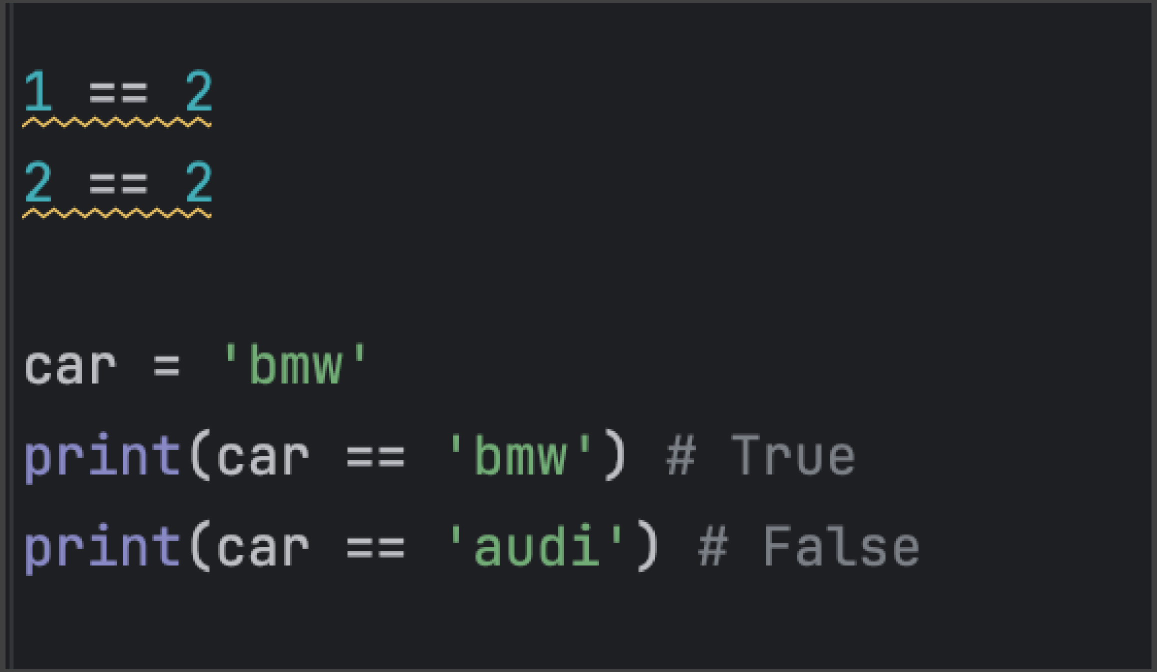


```
for x in []:
    print(x)
```

Review

Conditional Tests

- A conditional test is an expression that returns True or False.

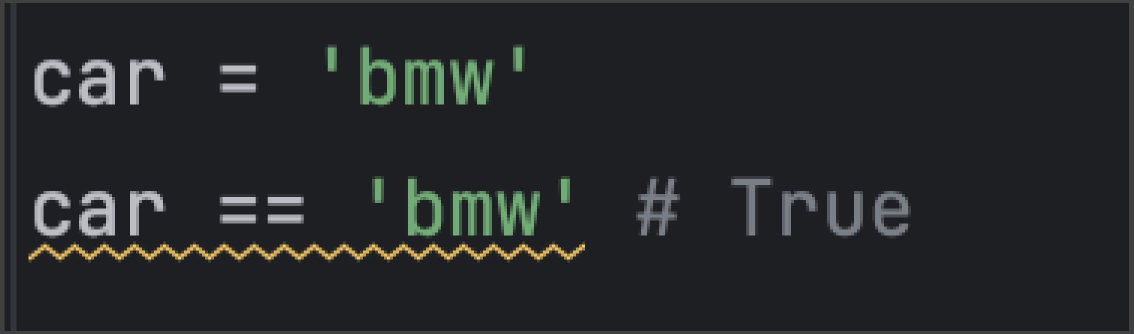


```
1 == 2
2 == 2

car = 'bmw'
print(car == 'bmw') # True
print(car == 'audi') # False
```

Equality (==)

- Check if two values are the same:

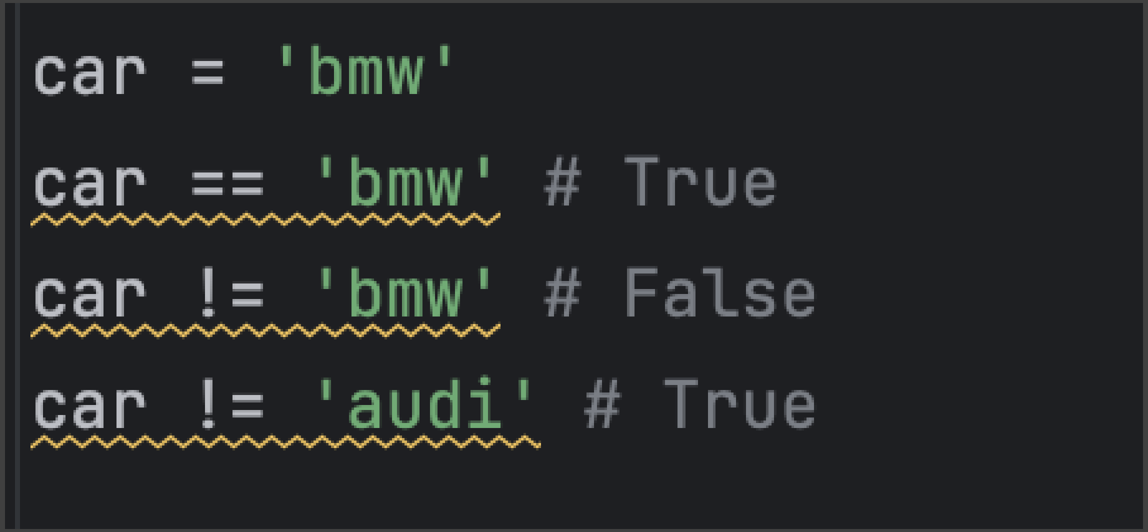


```
car = 'bmw'
car == 'bmw' # True
```

Inequality (!=)

REVIEW!

- Check if values are different:

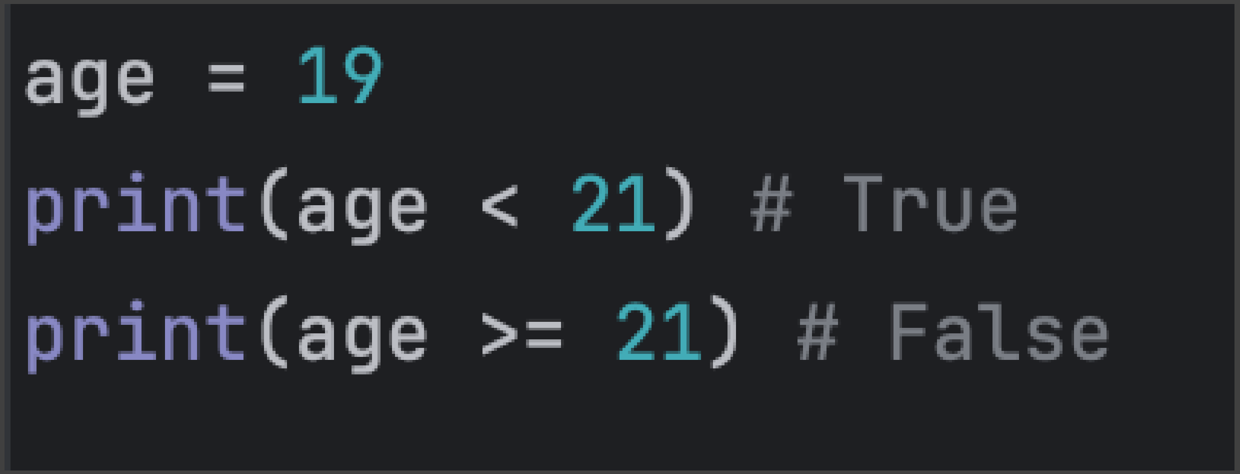


```
car = 'bmw'
car == 'bmw' # True
car != 'bmw' # False
car != 'audi' # True
```

Numerical Comparisons

REVIEW!

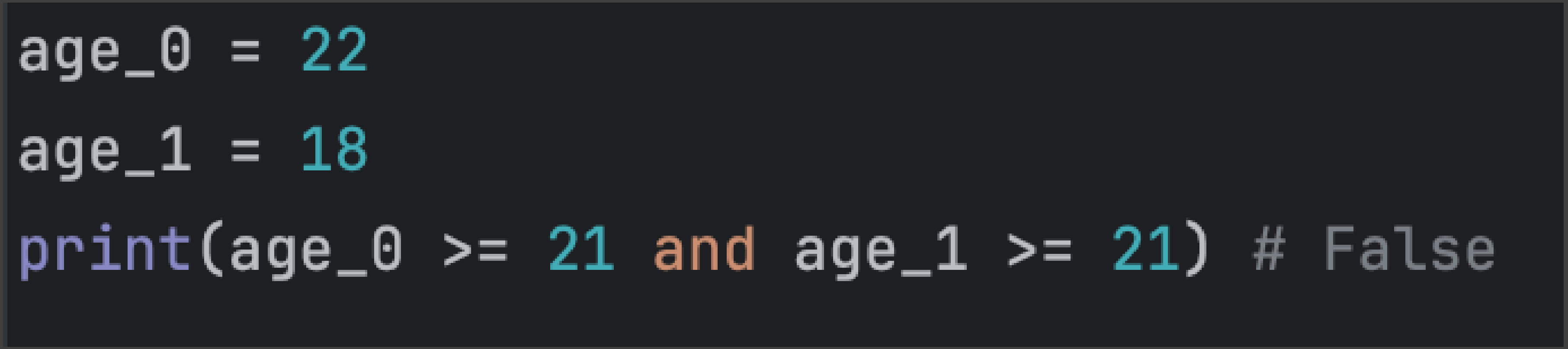
- Use comparison operators:



```python
age = 19
print(age < 21) # True
print(age >= 21) # False
```

Logical Operators: and

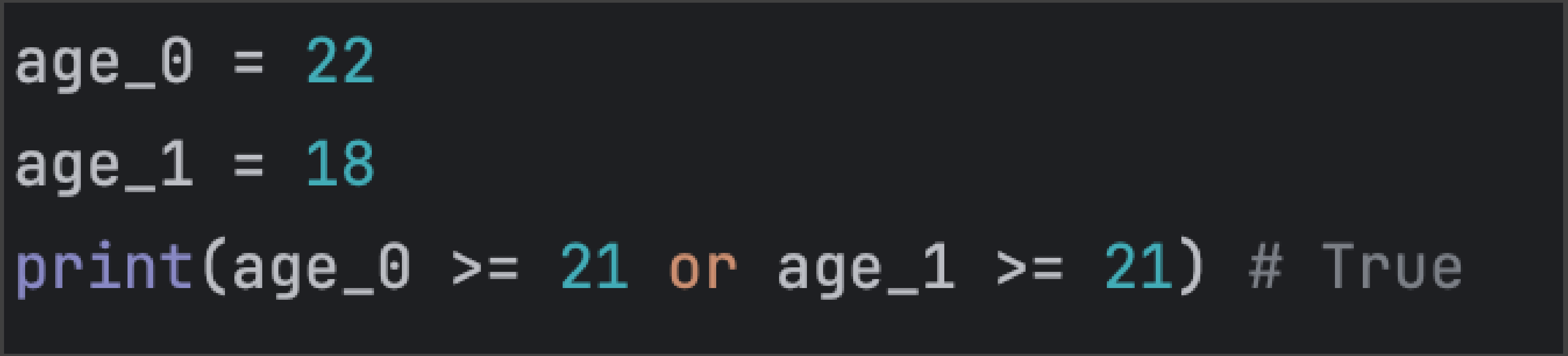
- Both conditions must be True:



```python
age_0 = 22
age_1 = 18
print(age_0 >= 21 and age_1 >= 21) # False
```

Logical Operators: or

- Only one condition needs to be True:

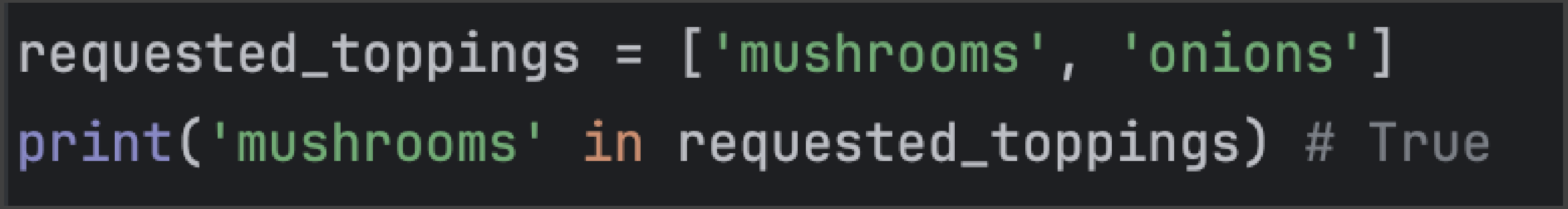


```python
age_0 = 22
age_1 = 18
print(age_0 >= 21 or age_1 >= 21) # True
```

Membership: in

REVIEW!

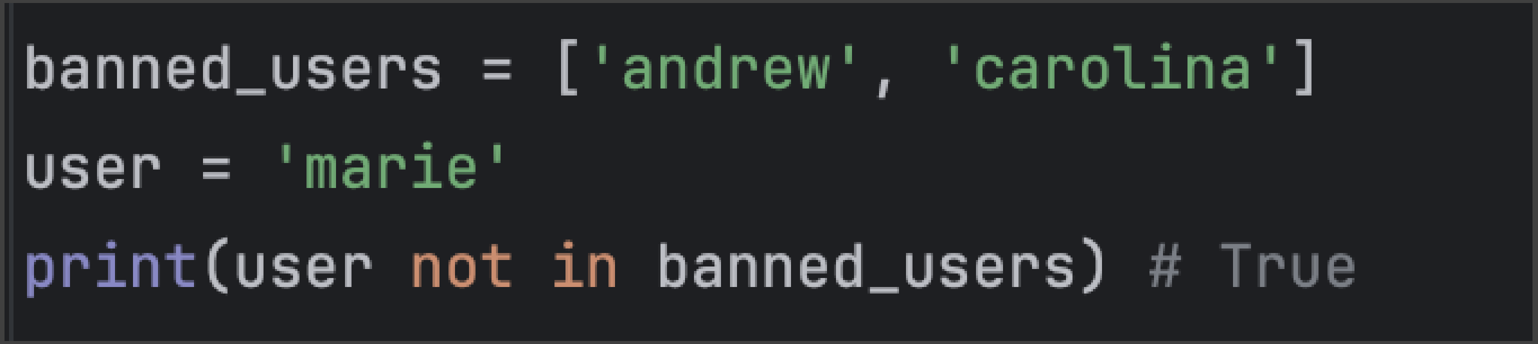
- Check if a value is in a list:



```python
requested_toppings = ['mushrooms', 'onions']
print('mushrooms' in requested_toppings) # True
```

Membership: not in

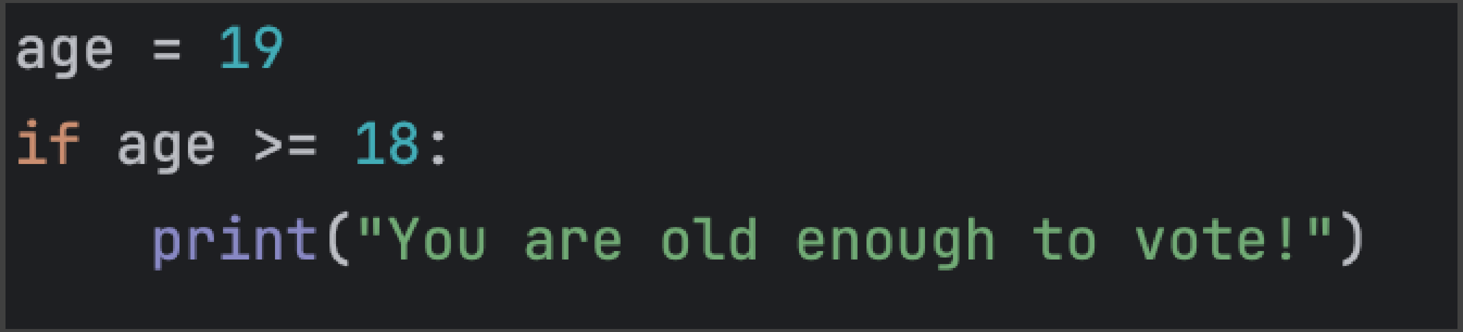
- Check if a value is not in a list:



```python
banned_users = ['andrew', 'carolina']
user = 'marie'
print(user not in banned_users) # True
```

if Statement

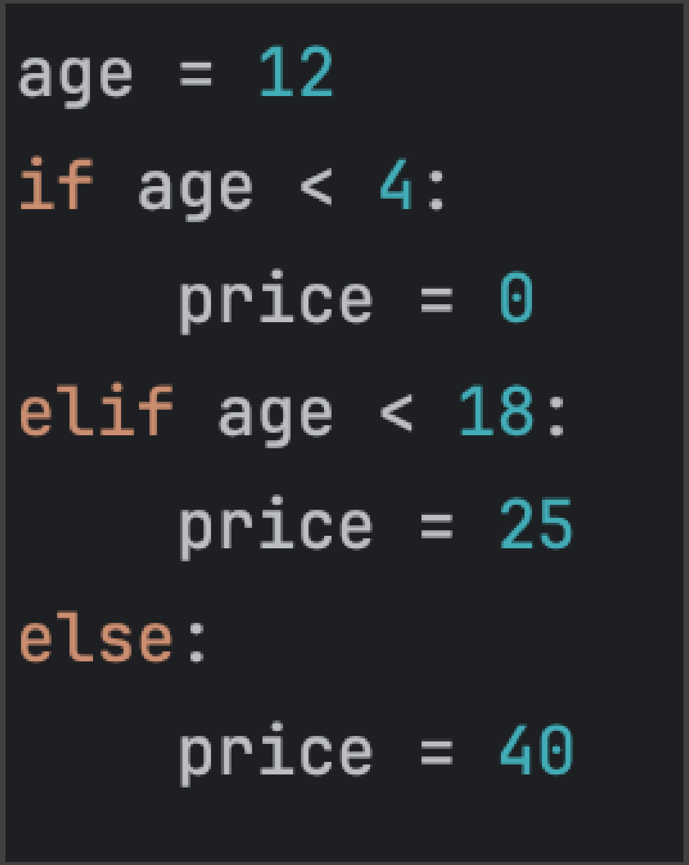
- Check if a condition is met and act accordingly



```
age = 19
if age >= 18:
    print("You are old enough to vote!")
```

if-elif-else Chains

- Check if a condition is met and act accordingly



```python
age = 12
if age < 4:
    price = 0
elif age < 18:
    price = 25
else:
    price = 40
```

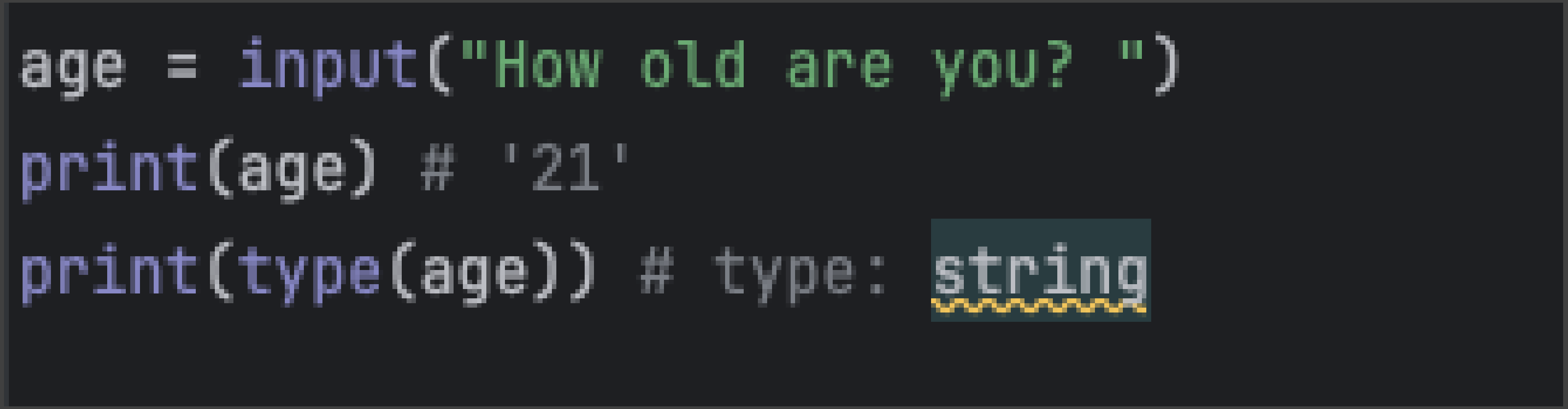
The input() Function

- input() shows a prompt to the user.
- Waits for the user to type and press Enter.
- Stores the response in a variable.

```python
message = input("Tell me something: ")
print(message)
```

Input is Always a String

- The input is always stored as a string

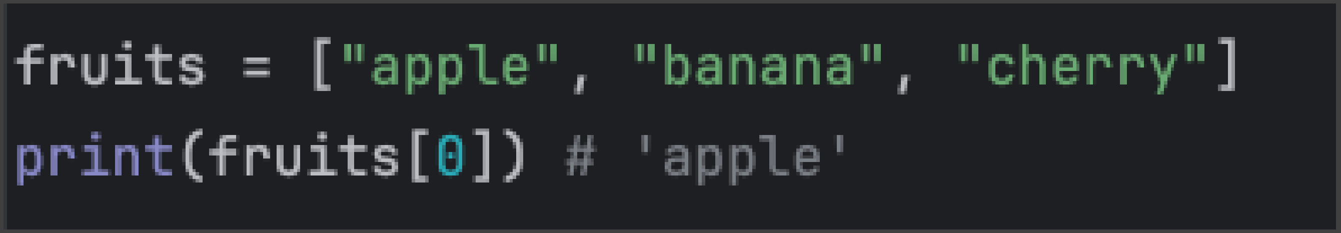


```
age = input("How old are you? ")
print(age) # '21'
print(type(age)) # type: string
```

Discovering Dictionaries

Why Dictionaries?

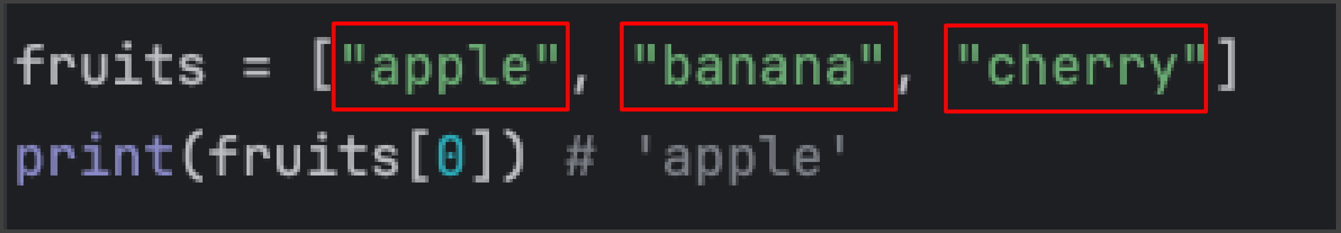
- Lists: store items by position.



```python
fruits = ["apple", "banana", "cherry"]
print(fruits[0]) # 'apple'
```

Why Dictionaries?

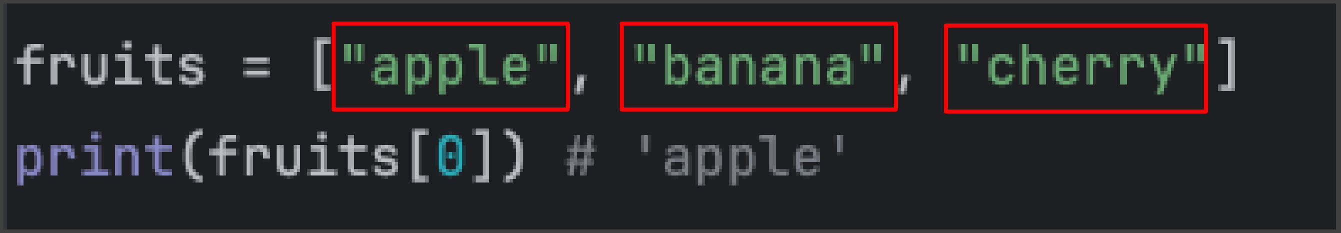
- Lists: store items by position.



```
fruits = ["apple", "banana", "cherry"]
print(fruits[0]) # 'apple'
```

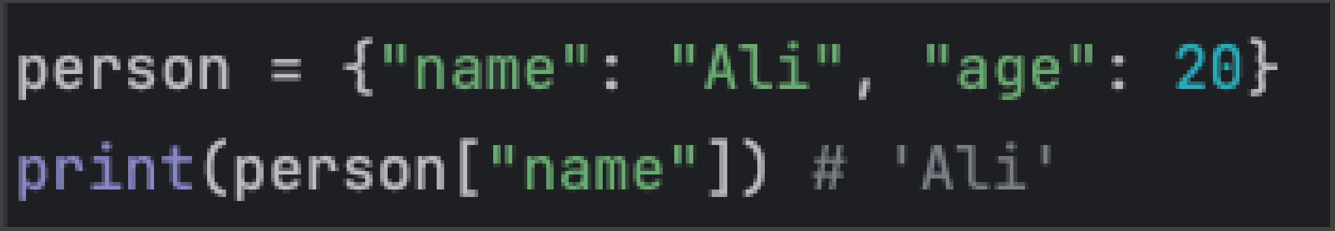
Why Dictionaries?

- Lists: store items by position.



```python
fruits = ["apple", "banana", "cherry"]
print(fruits[0]) # 'apple'
```

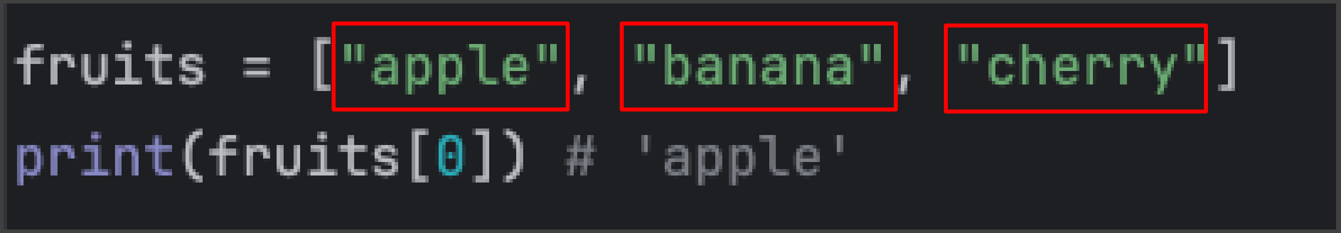
- Dictionaries: store items by name (key).



```python
person = {"name": "Ali", "age": 20}
print(person["name"]) # 'Ali'
```

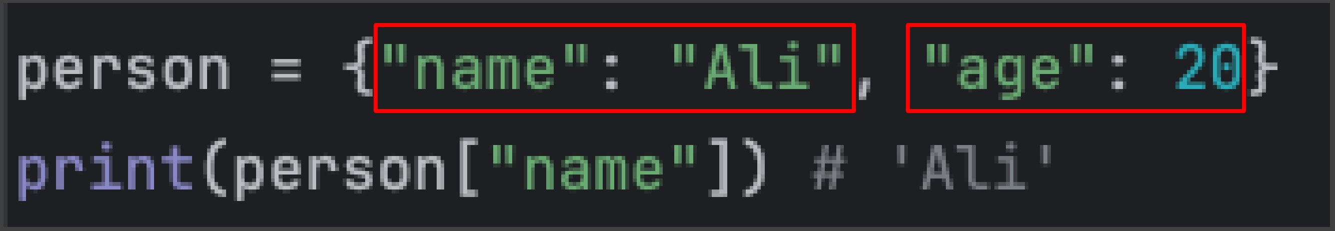
Why Dictionaries?

- Lists: store items by position.



```
fruits = ["apple", "banana", "cherry"]
print(fruits[0]) # 'apple'
```

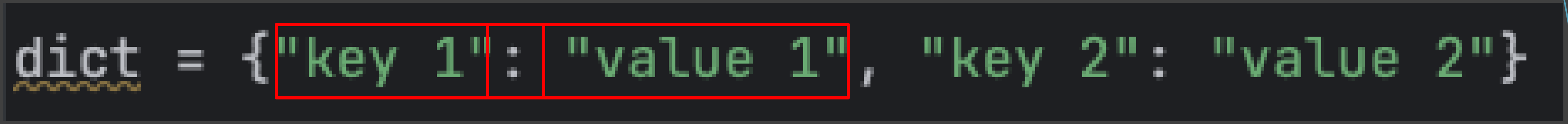
- Dictionaries: store items by name (key).



```
person = {"name": "Ali", "age": 20}
print(person["name"]) # 'Ali'
```

Dictionary Syntax

- A dictionary is wrapped in curly braces { }. Each key is connected to a value with a colon :. Multiple pairs are separated by commas.



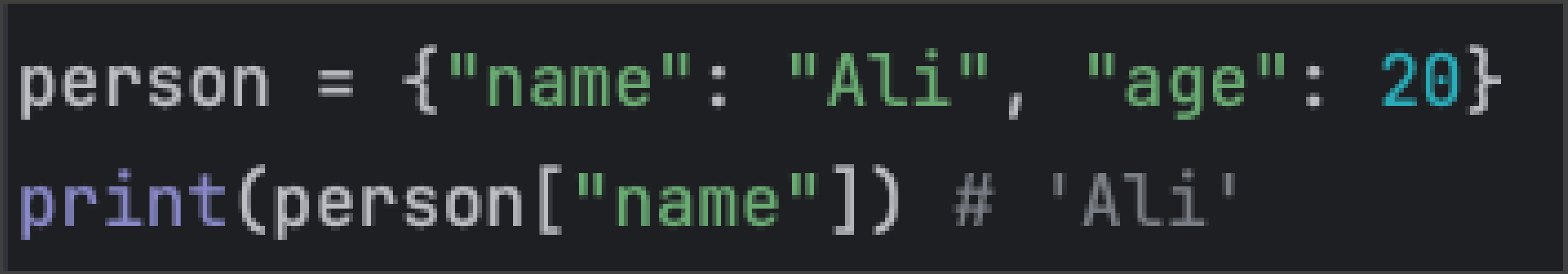
```
dict = {"key 1": "value 1", "key 2": "value 2"}
```

- Keys: must be unique.

- Values: can be numbers, strings, lists, or even other dictionaries.

Dictionary Syntax

- Values are accessed by their key.



```
person = {"name": "Ali", "age": 20}
print(person["name"]) # 'Ali'
```

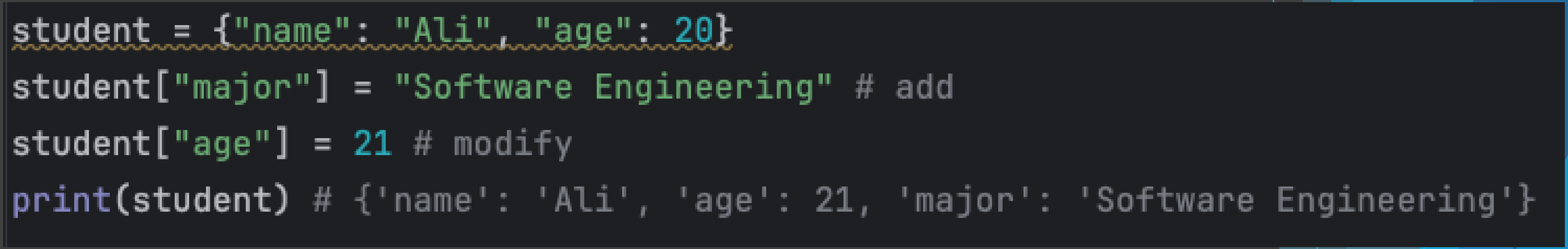
- There is no index!

Mini task 1

- Create a dictionary named student with the following keys:
  - name
  - age
- Then print both values in one sentence, e.g.:
  - “My name is Ali and I am 20 years old.”

Adding and Modifying Data

- Dictionaries are dynamic – you can add or change data anytime.



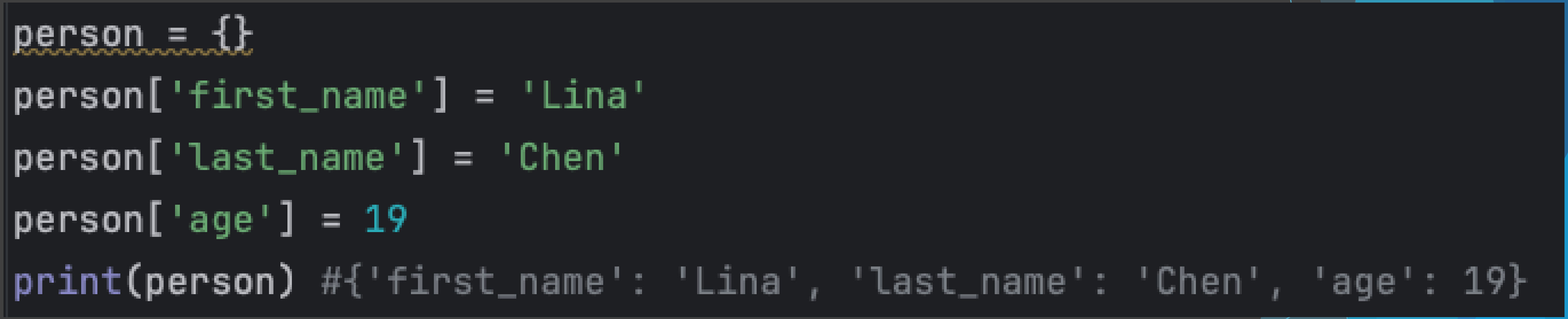
```python
student = {"name": "Ali", "age": 20}
student["major"] = "Software Engineering" # add
student["age"] = 21 # modify
print(student) # {'name': 'Ali', 'age': 21, 'major': 'Software Engineering'}
```

Mini task 2

- Add 3 more key-value pairs to your student dictionary.
- Print your dictionary.

Starting empty

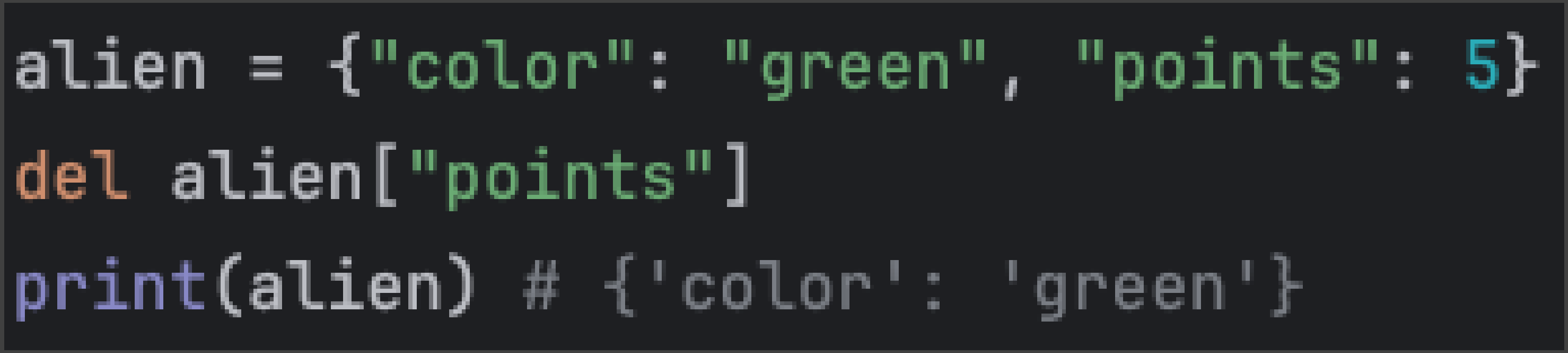
- You can start an empty dictionary and add values step by step.



```
person = {}
person['first_name'] = 'Lina'
person['last_name'] = 'Chen'
person['age'] = 19
print(person) #{'first_name': 'Lina', 'last_name': 'Chen', 'age': 19}
```

Deleting Key-Value Pairs

- Use del to permanently remove a pair.



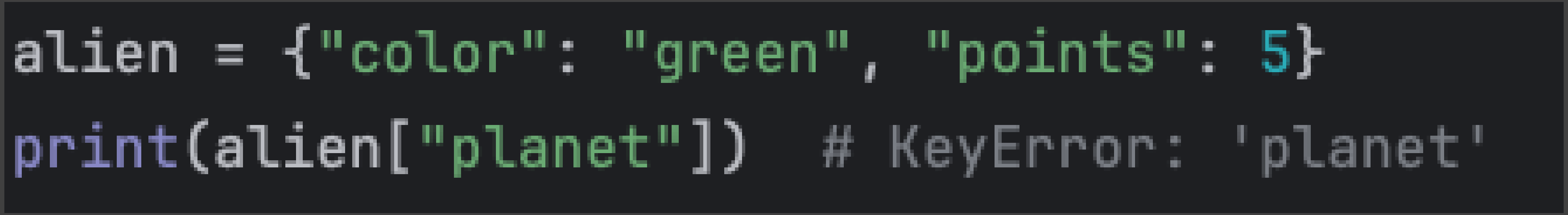
```python
alien = {"color": "green", "points": 5}
del alien["points"]
print(alien) # {'color': 'green'}
```

Mini task 3

- Delete one of the keys from your student dictionary.
- Print the dictionary.

Key error

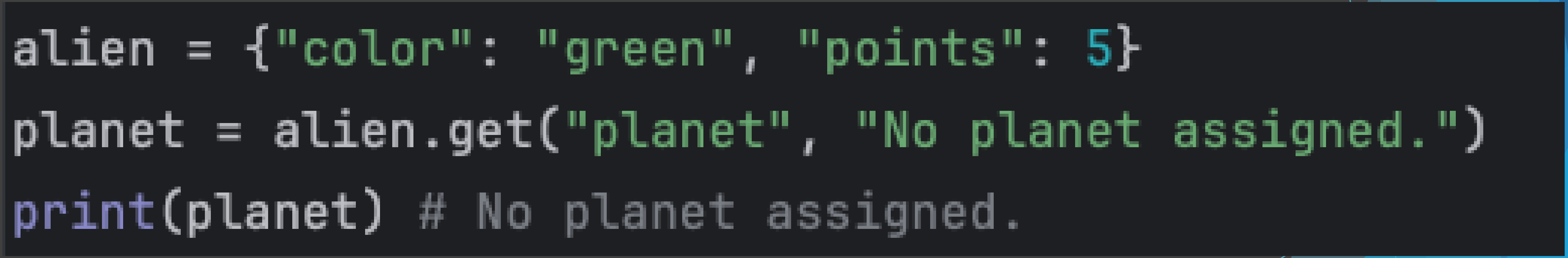
- Accessing a key that doesn’t exist results in key error.



```python
alien = {"color": "green", "points": 5}
print(alien["planet"])  # KeyError: 'planet'
```

Key error

- Accessing a key that doesn’t exist results in key error.
- Use .get() to safely access a key if you don’t know if it exists.



```python
alien = {"color": "green", "points": 5}
planet = alien.get("planet", "No planet assigned.")
print(planet) # No planet assigned.
```

Mini task 4

- Try to access a non-existing key from your dictionary using `.get()`. Provide your own default message.

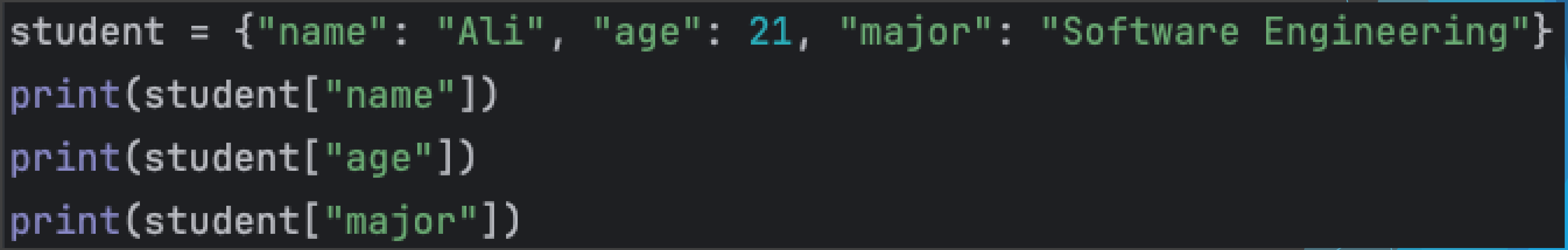
Wrap up

- Dictionaries store key-value pairs.
- Add or modify data with dictionary[key] = value.
- Delete data with del.
- Access safely using .get().

Looping through Dictionaries

From One Pair to Many

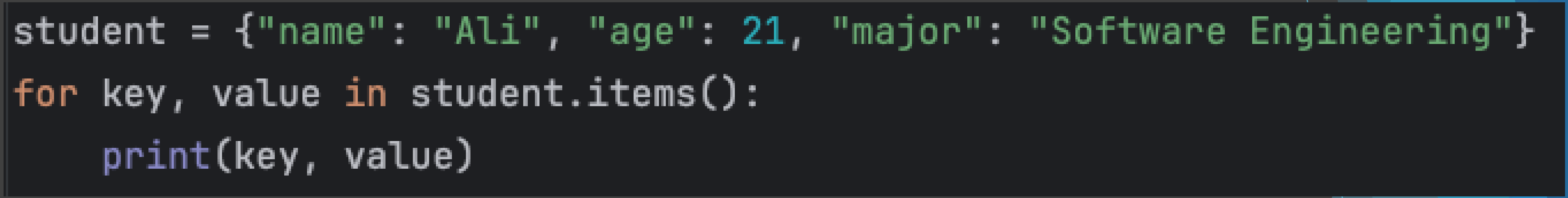
- We often want to see all key-value pairs in a dictionary.
- Instead of printing each key manually:



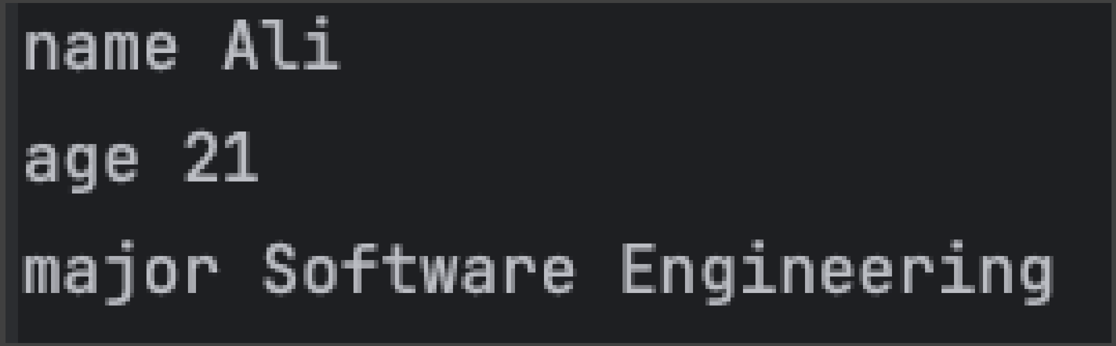
```python
student = {"name": "Ali", "age": 21, "major": "Software Engineering"}
print(student["name"])
print(student["age"])
print(student["major"])
```

Looping through key-value pairs

- Use .items() to get both the key and its value:



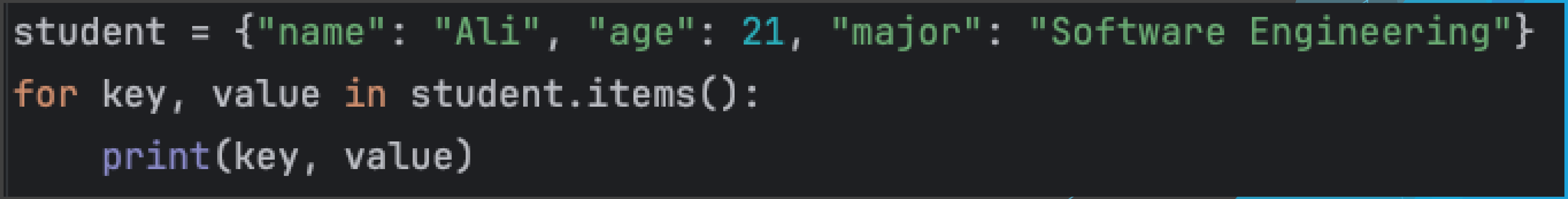
```python
student = {"name": "Ali", "age": 21, "major": "Software Engineering"}
for key, value in student.items():
    print(key, value)
```



```
name Ali
age 21
major Software Engineering
```

Mini task 5

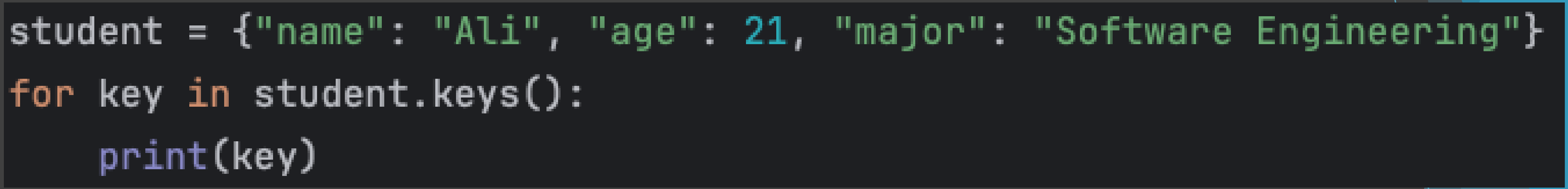
- Create a dictionary called favorite_foods with at least 3 people and their favorite foods.
- Loop through it and print sentences like:
    - “Sara likes pizza”
- Reminder:



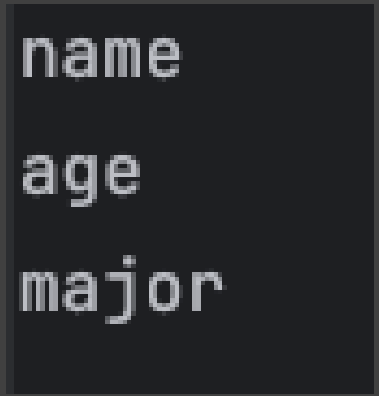
```python
student = {"name": "Ali", "age": 21, "major": "Software Engineering"}
for key, value in student.items():
    print(key, value)
```

Looping through keys only

- If you only need the keys, use .keys():



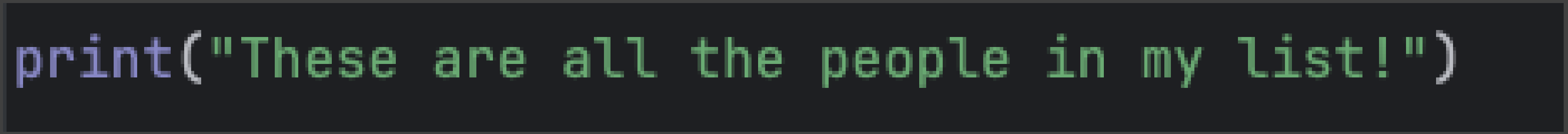
```python
student = {"name": "Ali", "age": 21, "major": "Software Engineering"}
for key in student.keys():
    print(key)
```



```
name
age
major
```

Mini task 6

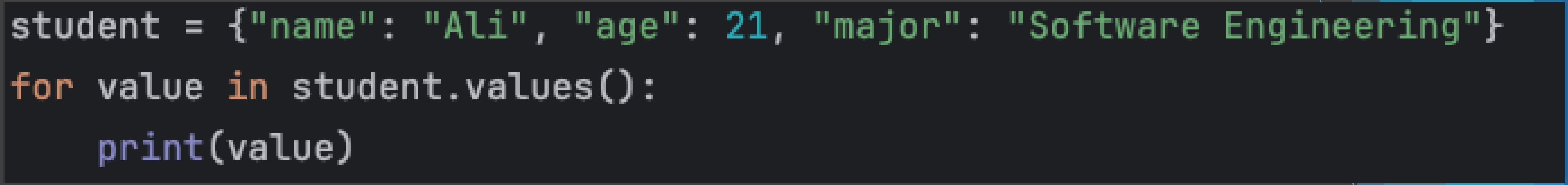
- Loop through your favorite_foods dictionary and print only the names (the keys).
- Then add one extra print statement:



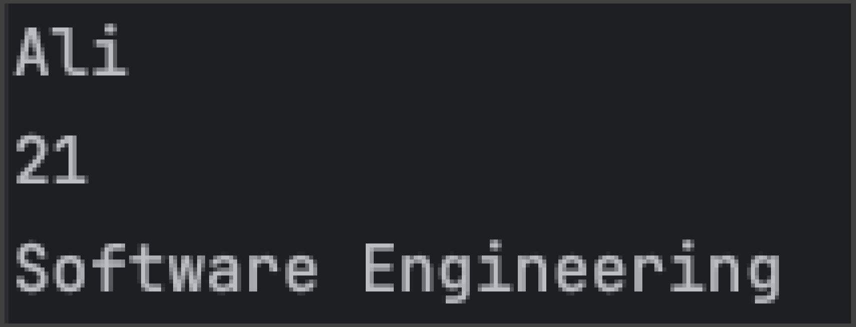
```python
print("These are all the people in my list!")
```

Looping through values only

- Use .values() to get all values:



```python
student = {"name": "Ali", "age": 21, "major": "Software Engineering"}
for value in student.values():
    print(value)
```

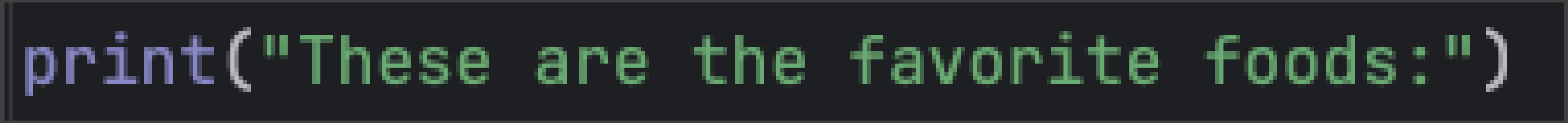


```
Ali
21
Software Engineering
```

Mini task 7

- Loop through your favorite_foods dictionary and print all foods only.

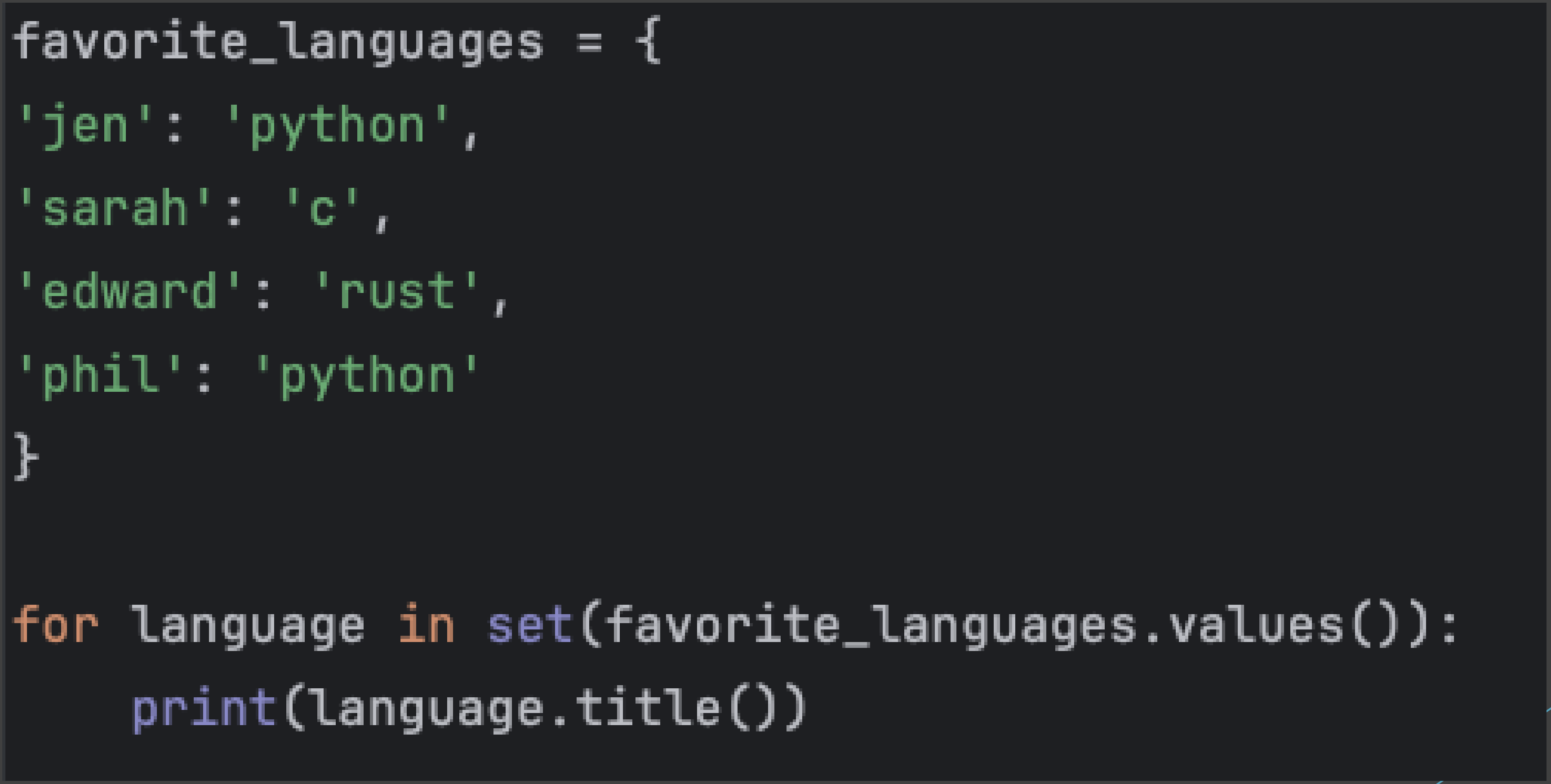
- Add a message before the loop:



```python
print("These are the favorite foods:")
```

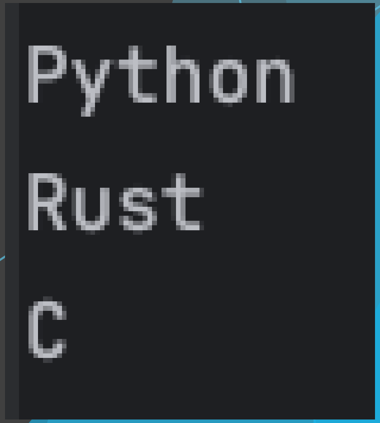
Avoiding Repeated Values with set()

- If some values repeat, we can make them unique using set():



```python
favorite_languages = {
'jen': 'python',
'sarah': 'c',
'edward': 'rust',
'phil': 'python'
}

for language in set(favorite_languages.values()):
    print(language.title())
```



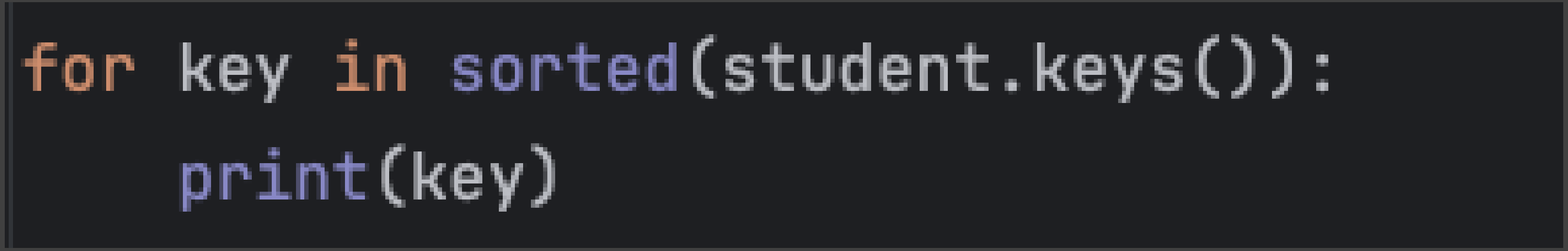
```
Python
Rust
C
```

Mini task 8

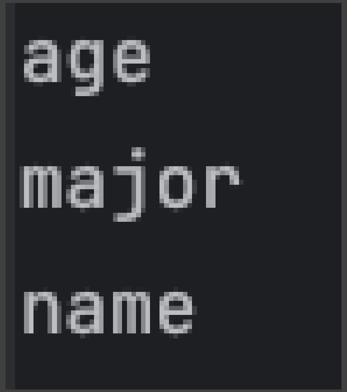
- Add one more friend to your favorite_foods dictionary, with the same food as another friend.
- Use set() to print all unique food values from the dictionary.

Sorting Dictionary Keys

- We can loop through keys in alphabetical order using sorted():



```python
for key in sorted(student.keys()):
    print(key)
```



```
age
major
name
```

Mini task 9

- Use your favorite_foods dictionary again.
- Loop through the names in sorted order and print them with their foods.
- Example:

```
Ali likes noodles.
Lina likes tacos.
Sara likes pizza.
```

Wrap up

- .items() – loop through key and value pairs.
- .keys() – loop through all keys.
- .values() – loop through all values.
- set() – remove duplicates.
- sorted() – order results alphabetically.

Exercises

Exercises

Task 1

Make a dictionary with 3 people and their favorite number.

Task 2

Print each person and their favorite number.

Task 3

Add one more person and their favorite number to the dictionary using assignment (=).

Task 4

Change the favorite number of one existing person.

Exercises

Task 5
Delete one person from the dictionary using del.

Task 6
Loop through all people and print their names only (keys).

Task 7
Loop through all favorite numbers only (values).

Task 8
Add a new entry where one person has more than one favorite number (a list of numbers).

Exercises

Task 9

Loop through the dictionary and print each person with all their favorite numbers.

Task 10

Add new information for each person: their country and major.
Now each person should have a small nested dictionary (person : {info}).

Task 11

Loop through the dictionary and print formatted sentences with each person’s name, country, and major.

Task 12

Add one more student with all details, then sort the names alphabetically and print them.