COL1000
Introduction to Programming

Priyanka Golia

Most (if not all) of the content is borrowed from Prof. Subodh Kumar's slides

Announcements

1. If you missed your minor exam due to medical reasons, please submit the medical certificate as soon as possible. Do not share it via email; follow the guidelines shared earlier.
2. We are receiving a large number of emails regarding lab attendance. Please refer to the announcement on Moodle for instructions on how to inform us. We will not respond to queries sent by email.
    - Only if you are absolutely sure that you arrived on time and informed the Lab TA about a system issue that prevented you from submitting your attendance, please follow the procedure mentioned in the Moodle announcement.
3. Lab Test 2 will begin on Monday, 22nd September.

Dictionaries

A dictionary in Python is like a real dictionary:

You look up a word (key) → you get its meaning (value).

In Python, it stores data as key–value pairs.

```
student = {
    "name": "Priya",
    "age": 18,
    "marks": [85, 92, 78]
}
```

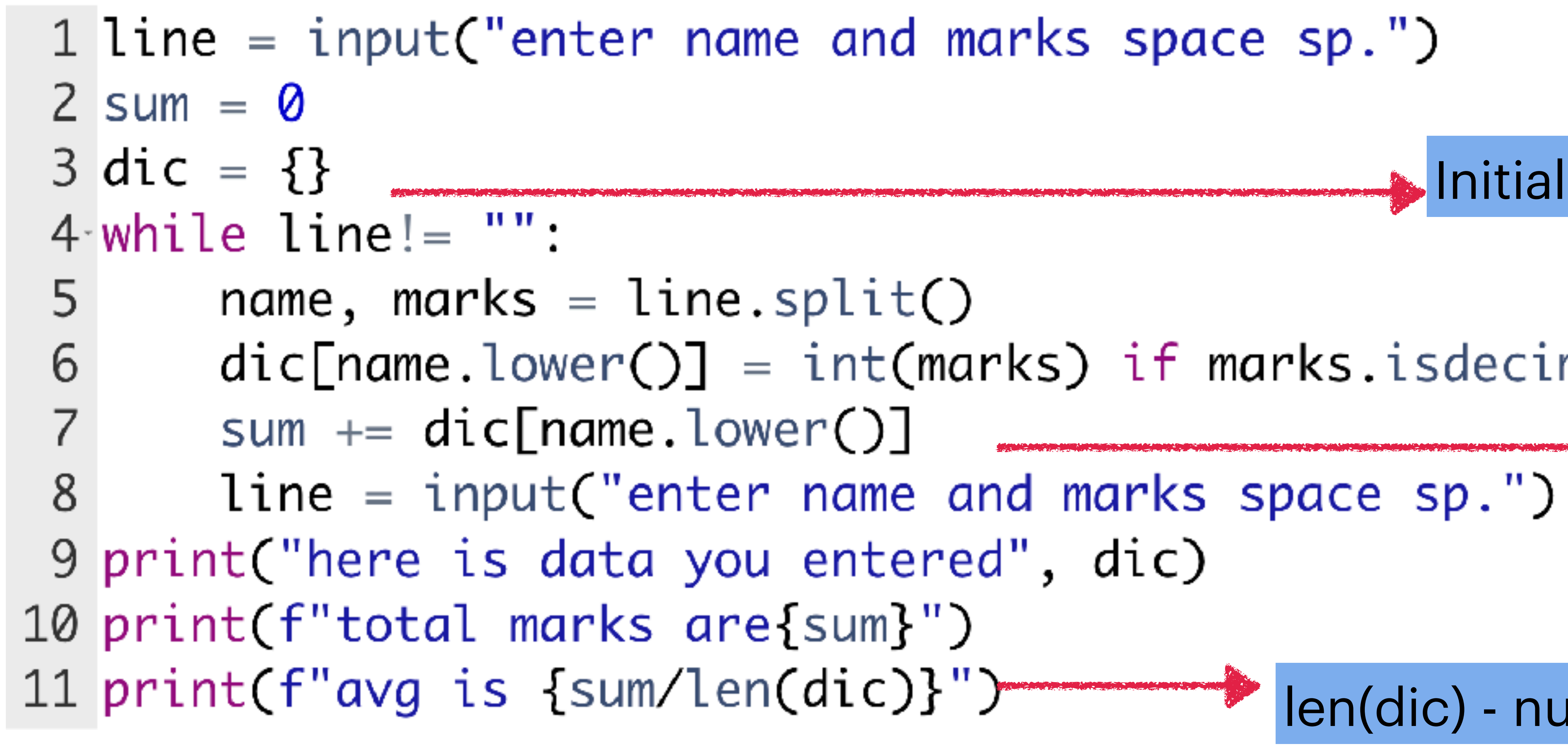
"name", "age", "marks" → keys

"Priya", 18, [85, 92, 78] → values

Keys must be unique.
Keys must be immutable —
Allowed as keys: int, float, str, **tuple**
Not allowed: **list**, dict, set

Values need not to be unique.
Values can mutate — it can be anything:
int, float, str, list, tuple, dictionary, set

Dictionaries



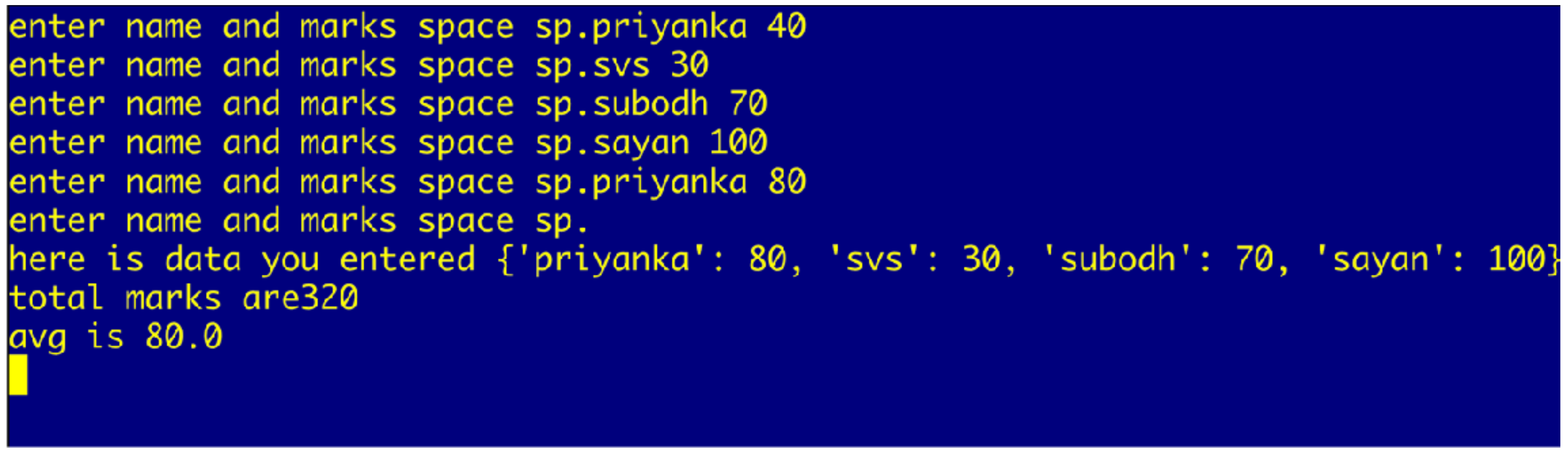
```python
line = input("enter name and marks space sp.")
sum = 0
dic = {}
while line!= "":
    name, marks = line.split()
    dic[name.lower()] = int(marks) if marks.isdecimal() else 0
    sum += dic[name.lower()]
    line = input("enter name and marks space sp.")
print("here is data you entered", dic)
print(f"total marks are{sum}")
print(f"avg is {sum/len(dic)}")
```

Initializing Dictionary

Key: name, value: marks

Accessing value by key

len(dic) - number of entries in dic



```
enter name and marks space sp.priyanka 40
enter name and marks space sp.svs 30
enter name and marks space sp.subodh 70
enter name and marks space sp.sayan 100
enter name and marks space sp.priyanka 80
enter name and marks space sp.
here is data you entered {'priyanka': 80, 'svs': 30, 'subodh': 70, 'sayan': 100}
total marks are320
avg is 80.0
```

Notice for key “Priyanka” is showing the latest marks entered. Keys have to unique and values are mutable, so it updates values to 80

Functions

A function is a block of code that performs a specific task.

Helps to reuse code without repeating.

Makes programs easier to read and debug.

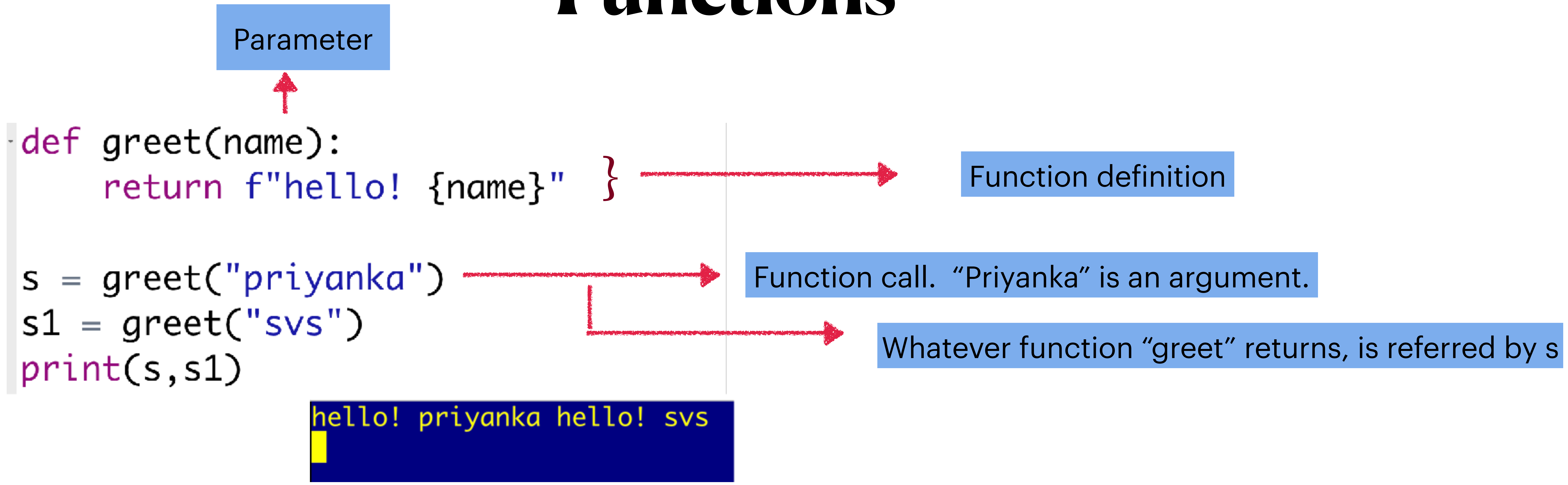
```
Output = a * b + c * d
output = f(a,b) + g(c,d)
```

f(a,b) — takes a,b, computes a*b, and return the outcome.

g(c,d) — takes c,d, computes c*d, and return the outcome.

Formally: A function is a reusable block of code that takes inputs (arguments), performs a task, and optionally returns a value.

Functions



```
def greet(name):
    return f"hello! {name}"

s = greet("priyanka")
s1 = greet("svs")
print(s,s1)
```

Function call. “Priyanka” is an argument.

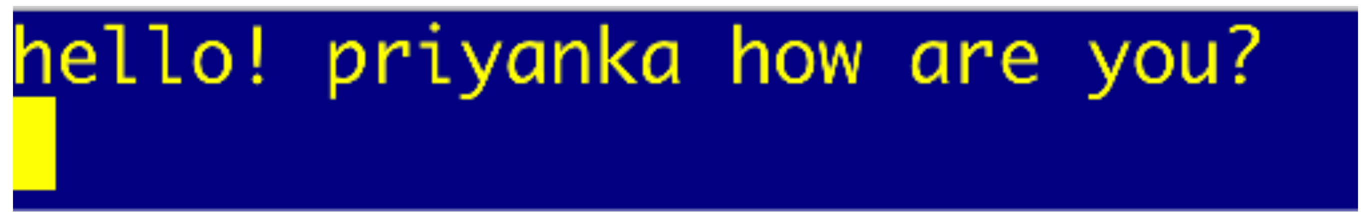
Whatever function “greet” returns, is referred by s

```
hello! priyanka hello! svs
```

- **def** starts a function definition.
- greet is the **function name** (same rules as variable names).
- (name) are the **parameters**.
- **:** ends the header line; the indented block is the function body.
- **return** sends a value back to the caller (optional).

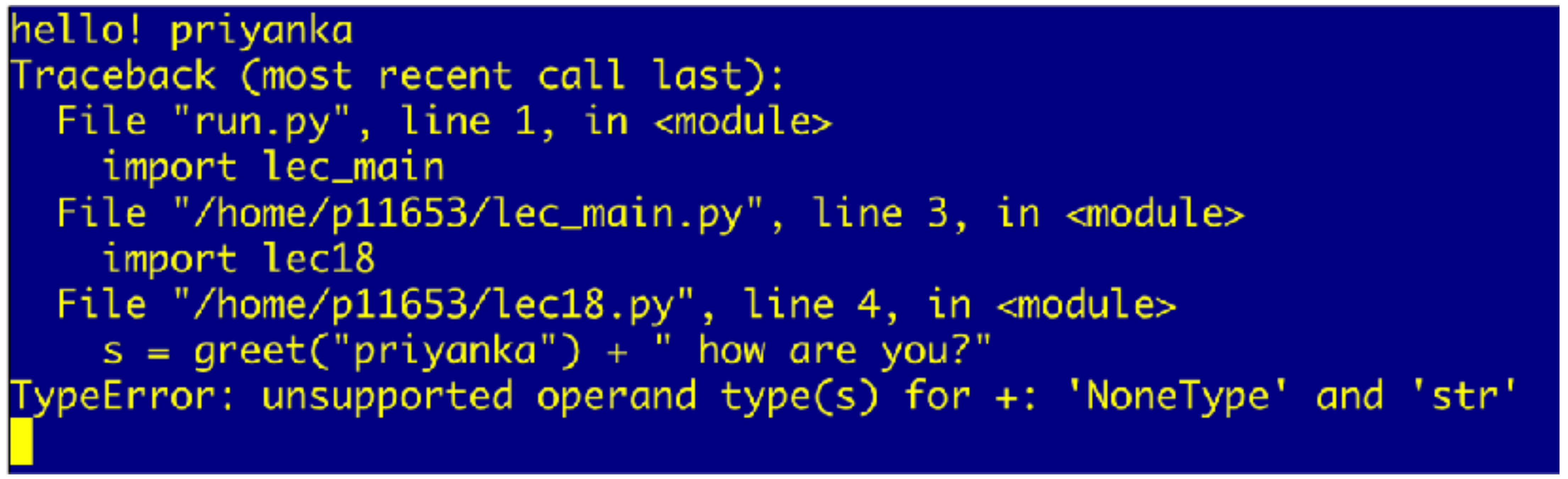
Functions

```python
def greet(name):
    return f"hello! {name}"

s = greet("priyanka") + " how are you?"
print(s)
```



```
hello! priyanka how are you?
```

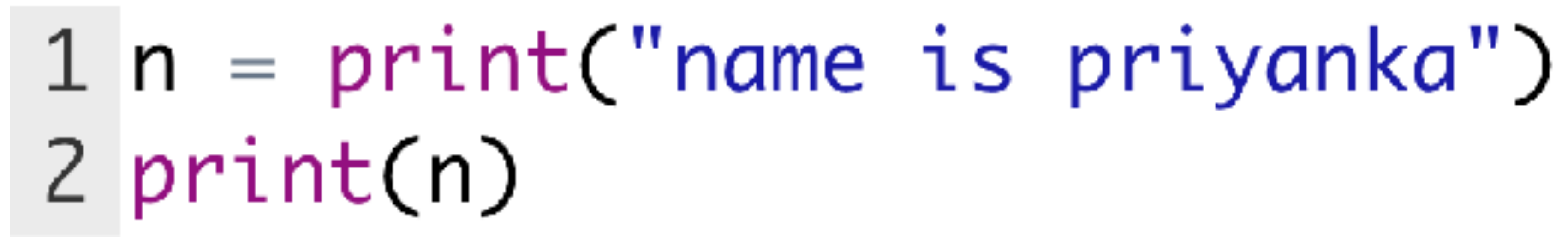
```python
def greet(name):
    print(f"hello! {name}")

s = greet("priyanka") + " how are you?"
print(s)
```



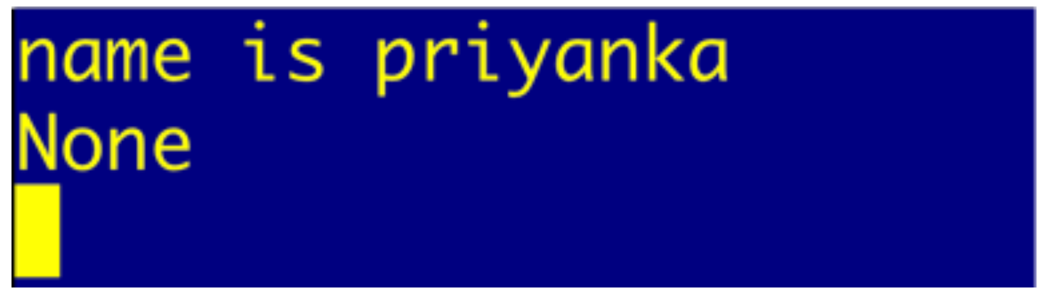
```
hello! priyanka
Traceback (most recent call last):
  File "run.py", line 1, in <module>
    import lec_main
  File "/home/p11653/lec_main.py", line 3, in <module>
    import lec18
  File "/home/p11653/lec18.py", line 4, in <module>
    s = greet("priyanka") + " how are you?"
TypeError: unsupported operand type(s) for +: 'NoneType' and 'str'
```

You can use explicit return statements with or without a return value. If you build a return statement without specifying a return value, then you'll be implicitly returning None.

Functions

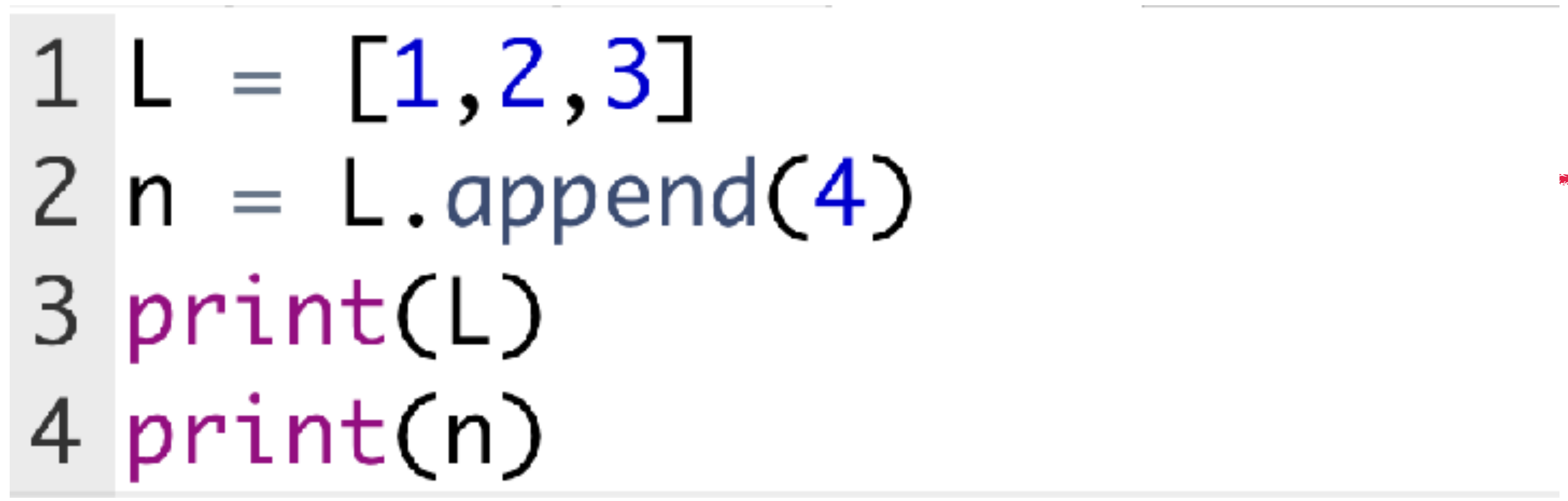


```python
n = print("name is priyanka")
print(n)
```

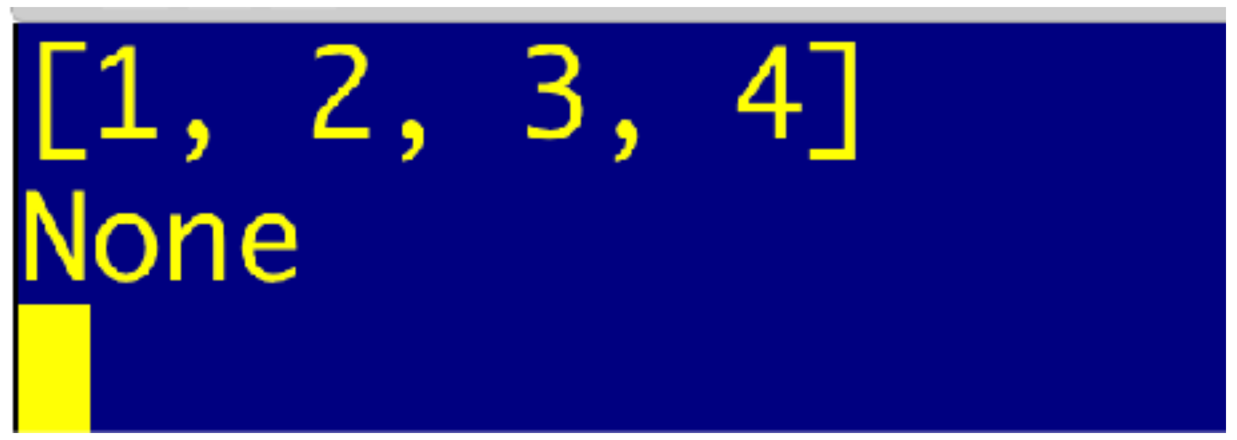


```
name is priyanka
None
```

Is there any error or is it correct?

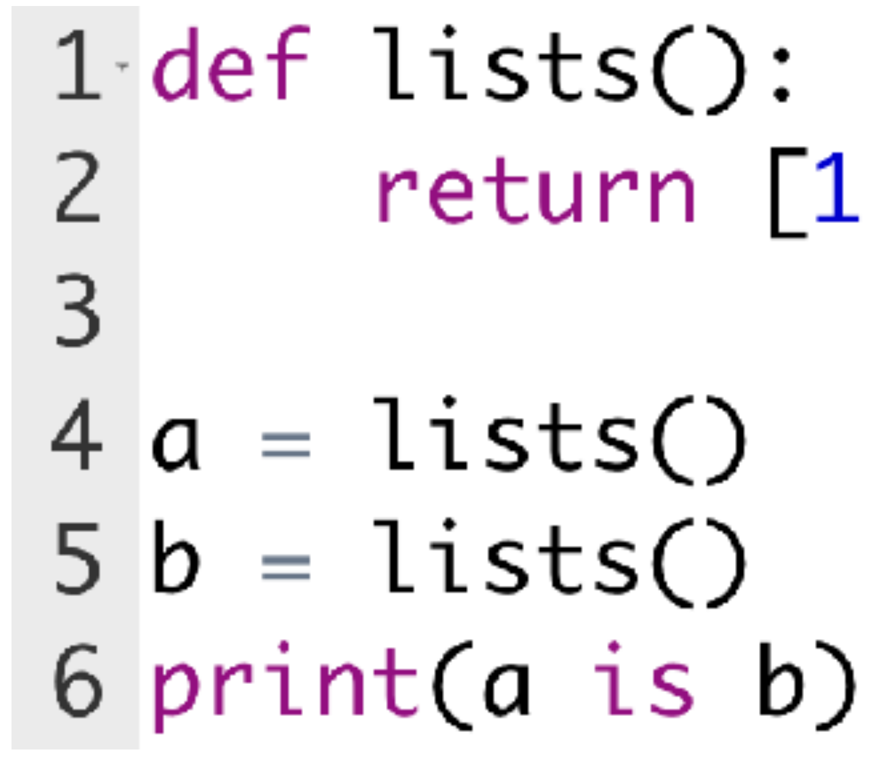


```python
L = [1,2,3]
n = L.append(4)
print(L)
print(n)
```

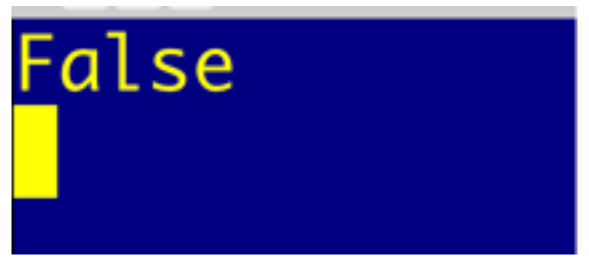


```
[1, 2, 3, 4]
None
```

Functions

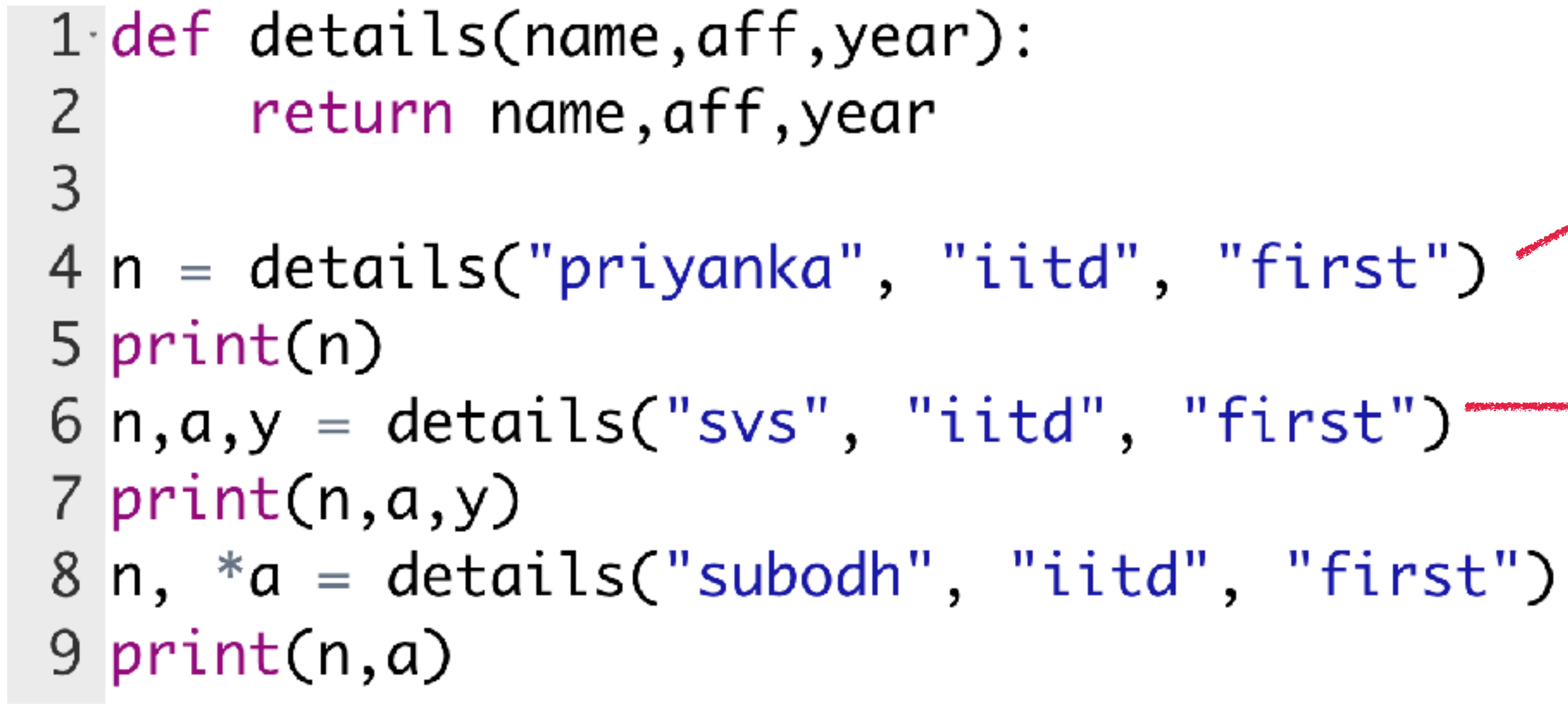


```python
def lists():
    return [1,2,3]

a = lists()
b = lists()
print(a is b)
```



```
False
```

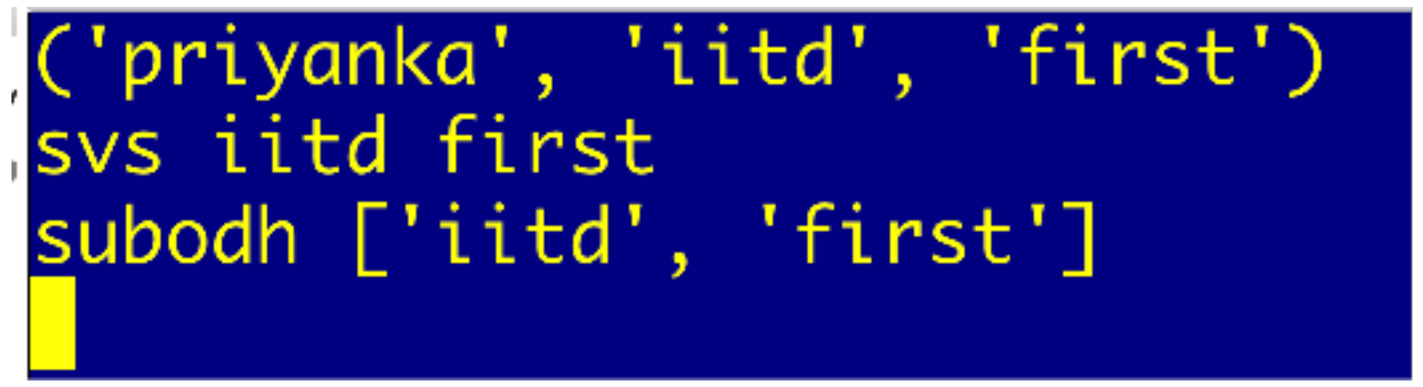
Return creates a new object everytime we call the functions, and return the reference.



```python
def details(name,aff,year):
    return name,aff,year

n = details("priyanka", "iitd", "first")
print(n)
n,a,y = details("svs", "iitd", "first")
print(n,a,y)
n, *a = details("subodh", "iitd", "first")
print(n,a)
```

Function can return reference to only one object, so multiple values are “packed” into a tuple automatically.

Now, in right hand side, we have a tuple object. All the rules of advance assignments apply here!



```
('priyanka', 'iitd', 'first')
svs iitd first
subodh ['iitd', 'first']
```