ReDI School of Digital Integration

Introduction to Data Analytics Spring 2023

Lecture 2

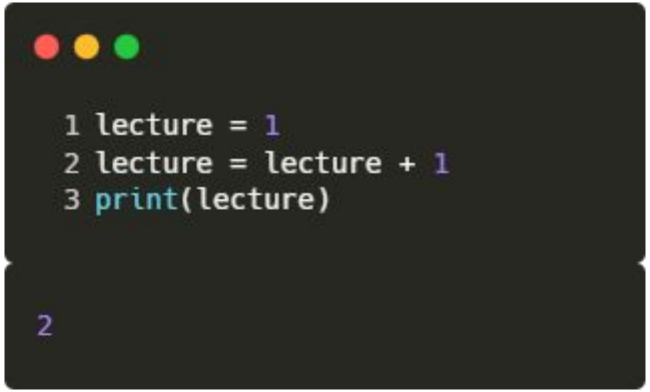
We use tech to connect human potential and opportunity with dignity & humility

Today's topics

- Basics of Programming
- Control Structures
  - Conditionals (if/else)
  - Loops
- Functions
- Working with files

Basics of Programming (Quiz)

Basics - Variables

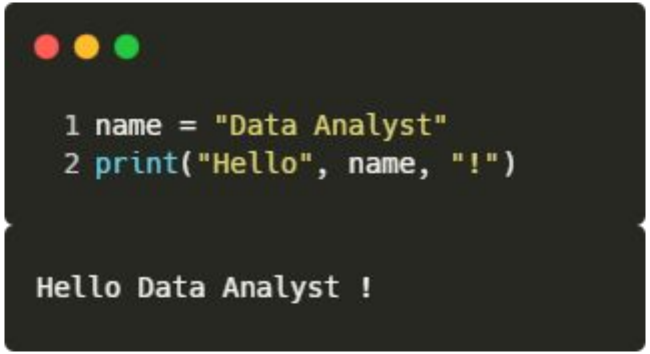


```
lecture = 1
lecture = lecture + 1
print(lecture)
```

```
2
```

- Can be named whatever you want
- Contain a value that can be overwritten

Basics - Print()

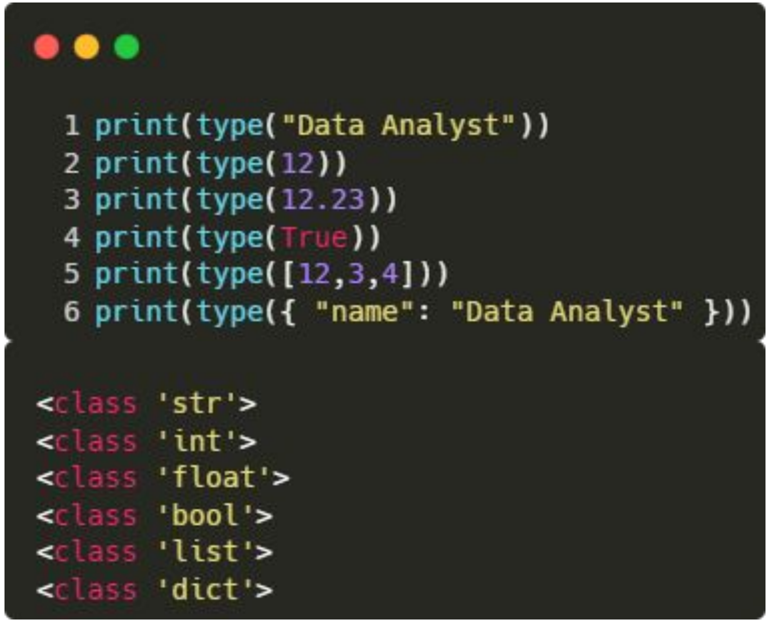


```python
name = "Data Analyst"
print("Hello", name, "!")
```

```
Hello Data Analyst !
```

- Prints strings, variables, etc. on the screen.
- You can pass multiple variables with comma separation.

Basics - Data Types

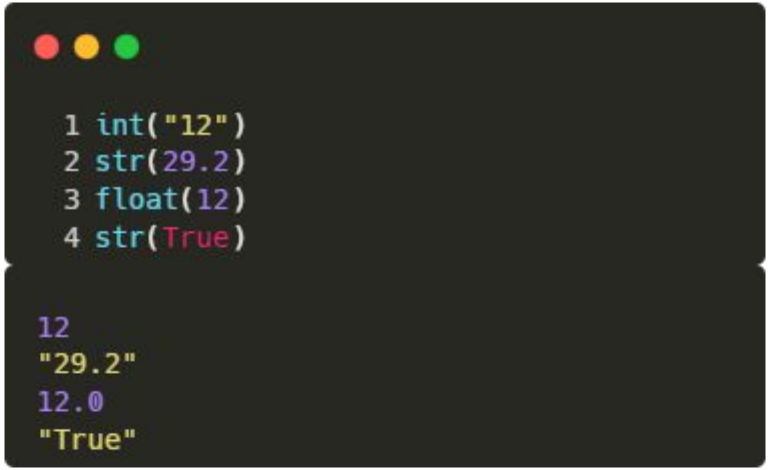


```
print(type("Data Analyst"))
print(type(12))
print(type(12.23))
print(type(True))
print(type([12,3,4]))
print(type({ "name": "Data Analyst" }))
```

```
<class 'str'>
<class 'int'>
<class 'float'>
<class 'bool'>
<class 'list'>
<class 'dict'>
```

- Tells us, what type a value is
- ***int*** and ***float*** can be used for mathematical operations
- ***str*** is text
- ***bool*** is useful for decision making
- ***list*** and ***dict*** contain multiple values

Basics - Type conversion

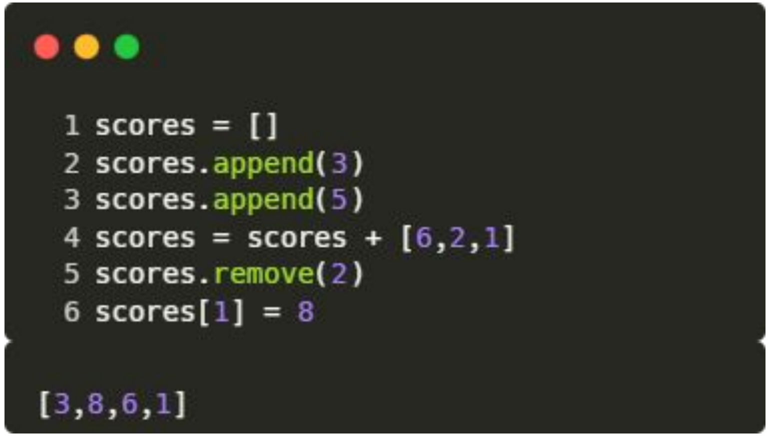


```
int("12")
str(29.2)
float(12)
str(True)
```

```
12
"29.2"
12.0
"True"
```

- Some types can be converted into other types
- Since mathematical operations can only be performed on numbers, you may need to convert a string containing a number into an actual number (int/float)

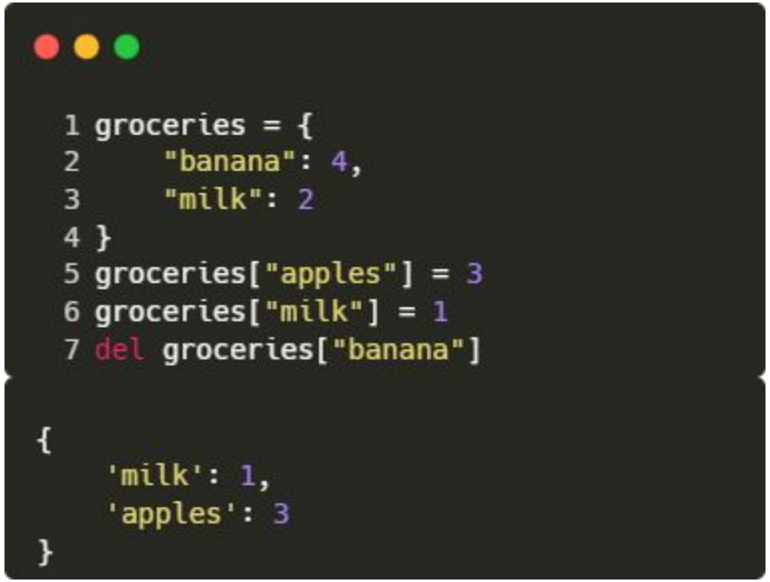
Basics - Lists


```
scores = []
scores.append(3)
scores.append(5)
scores = scores + [6,2,1]
scores.remove(2)
scores[1] = 8
```

```
[3,8,6,1]
```

- Lists are containers with multiple values
- You can identify lists by ***square brackets***
- Elements can be added, removed, and replaced
- More functions exist such as sort, reverse, clear, count, etc.
- Elements can be accessed by the ***index***.

Basics - Dictionaries

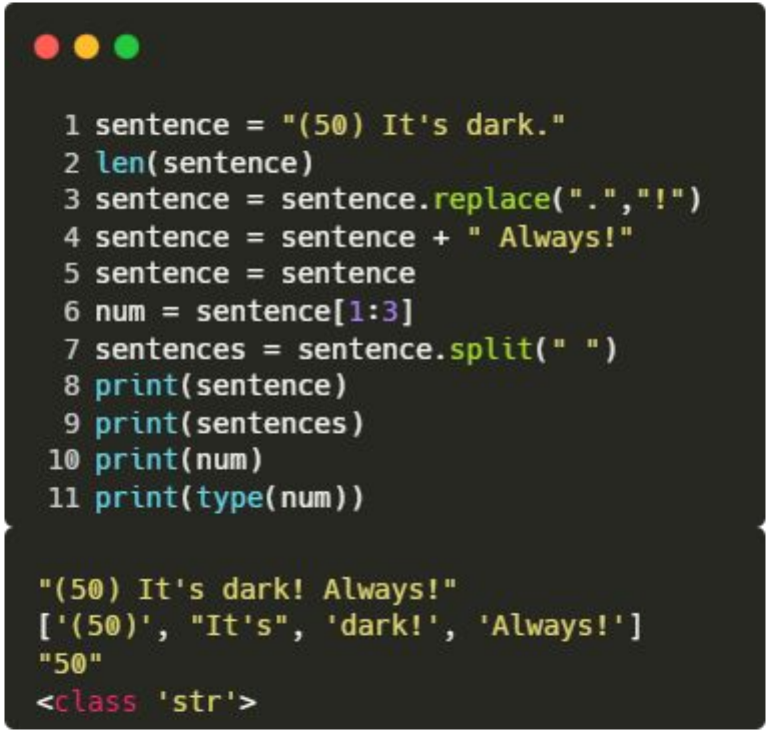


```
groceries = {
    "banana": 4,
    "milk": 2
}
groceries["apples"] = 3
groceries["milk"] = 1
del groceries["banana"]
```

```
{
    'milk': 1,
    'apples': 3
}
```

- Dictionaries are containers with key/value pairs
- The keys are strings
- The values can be anything. Even another dictionary!
- You can identify dictionaries by their **curly braces**
- Elements can be added, removed, and replaced
- Elements can be accessed by the **key**.

Basics - Strings



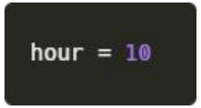
```
sentence = "(50) It's dark."
len(sentence)
sentence = sentence.replace(".","!")
sentence = sentence + " Always!"
sentence = sentence
num = sentence[1:3]
sentences = sentence.split(" ")
print(sentence)
print(sentences)
print(num)
print(type(num))
```

```
"(50) It's dark! Always!"
['(50)', "It's", 'dark!', 'Always!']
"50"
<class 'str'>
```

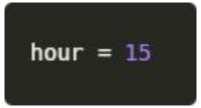
- Strings are a sequence of characters, marked **inside two quotes**.
- Strings can be analyzed, manipulated, split, combined, etc.

Control Structures: Conditionals (if/else)

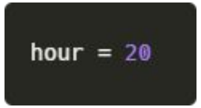
Control structures: Conditionals (if/else)



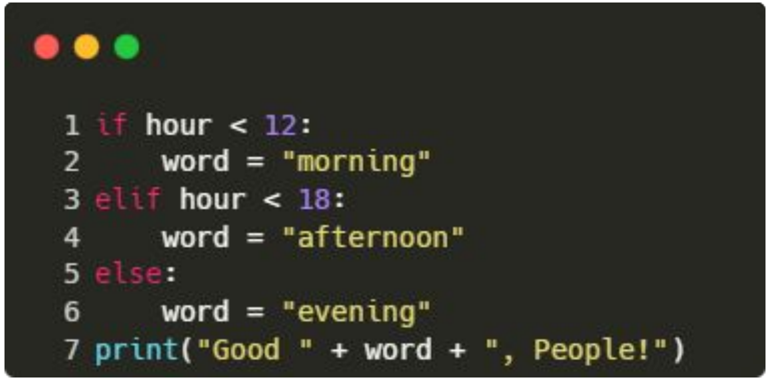
```
hour = 10
```



```
hour = 15
```



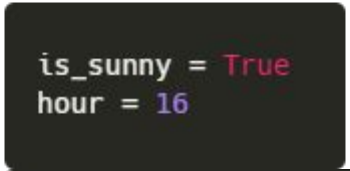
```
hour = 20
```



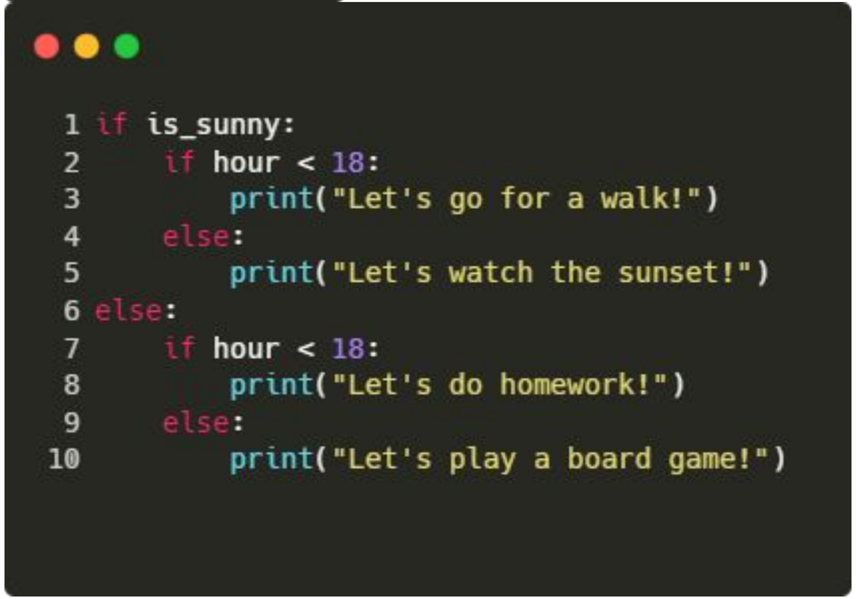
```python
if hour < 12:
    word = "morning"
elif hour < 18:
    word = "afternoon"
else:
    word = "evening"
print("Good " + word + ", People!")
```

- Depending on a **condition**, you can execute different code.
- The condition has to result in a **bool**.

Control structures: Nested Conditionals (if/else)



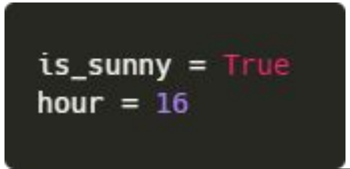
```python
is_sunny = True
hour = 16
```



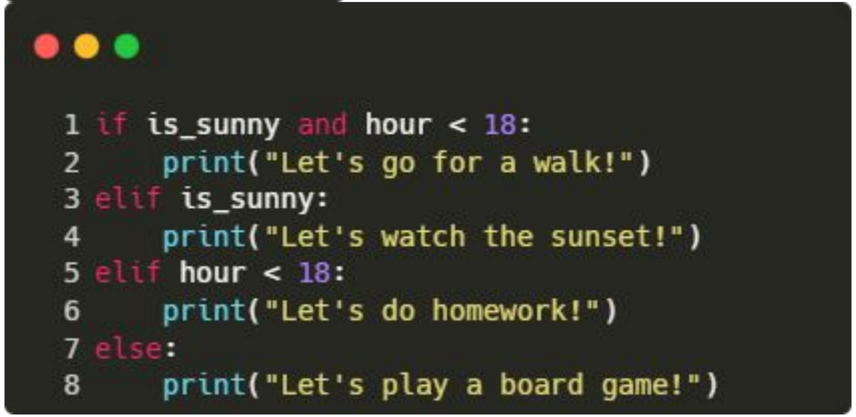
```python
if is_sunny:
    if hour < 18:
        print("Let's go for a walk!")
    else:
        print("Let's watch the sunset!")
else:
    if hour < 18:
        print("Let's do homework!")
    else:
        print("Let's play a board game!")
```

- Depending on a **condition**, you can execute different code.
- The condition has to result in a **bool**.
- The condition can be a **bool variable** or a **comparison**.
- Conditions can be **nested**.

Control structures: Combined Conditionals (if/else)



```python
is_sunny = True
hour = 16
```

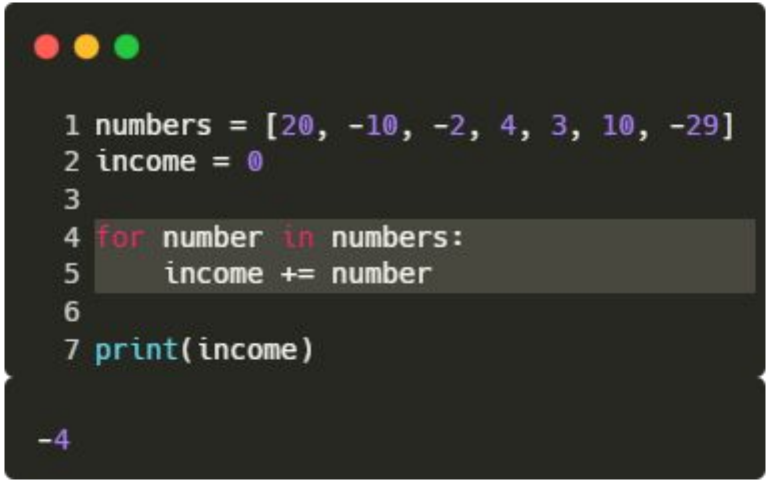


```python
if is_sunny and hour < 18:
    print("Let's go for a walk!")
elif is_sunny:
    print("Let's watch the sunset!")
elif hour < 18:
    print("Let's do homework!")
else:
    print("Let's play a board game!")
```

- Depending on a **condition**, you can execute different code.
- The condition has to result in a **bool**.
- The condition can be a **bool variable** or a **comparison**.
- Conditions can be **nested**.
- Conditions can be **combined**.

Control Structures: Loops

Control structures: Loops

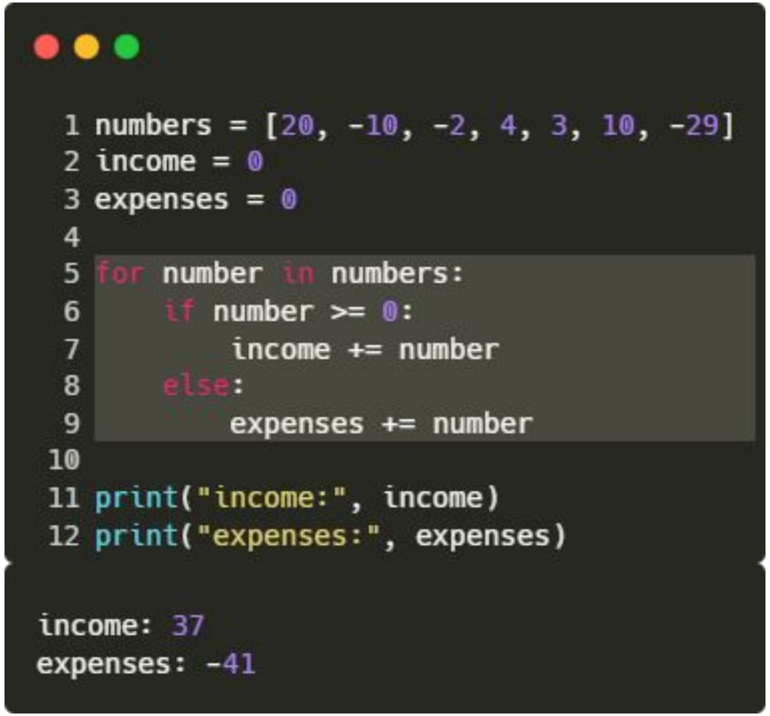


```
numbers = [20, -10, -2, 4, 3, 10, -29]
income = 0

for number in numbers:
    income += number

print(income)
```

```
-4
```

- Loops allow you to run the same code for a sequence (list).
- A for loop executes the inside of the for loop for every element in a sequence (list).
- It creates a temporary variable that is named by you.

Control structures: Loops



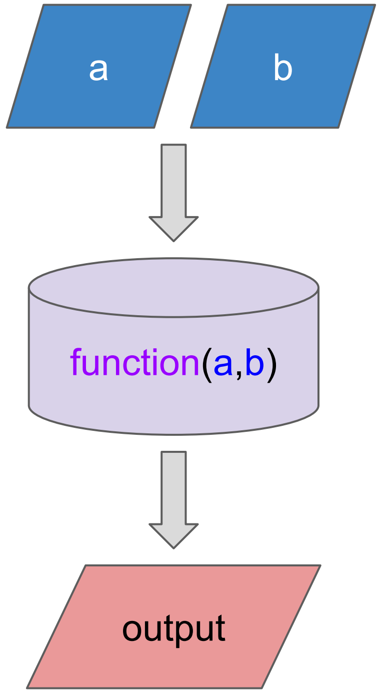
```
numbers = [20, -10, -2, 4, 3, 10, -29]
income = 0
expenses = 0

for number in numbers:
    if number >= 0:
        income += number
    else:
        expenses += number

print("income:", income)
print("expenses:", expenses)
```

```
income: 37
expenses: -41
```

- Loops allow you to run the same code for a sequence (list).
- A for loop executes the inside of the for loop for every element in a sequence (list).
- It creates a temporary variable that is named by you.
- For loops are often combined with if/else, but the code can be anything; even another for loop!

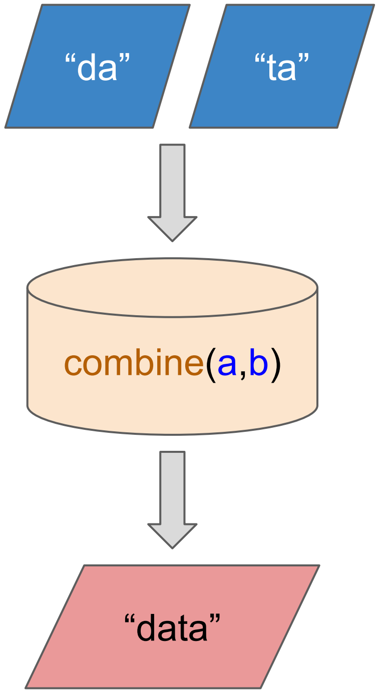
Functions

Functions: Concept



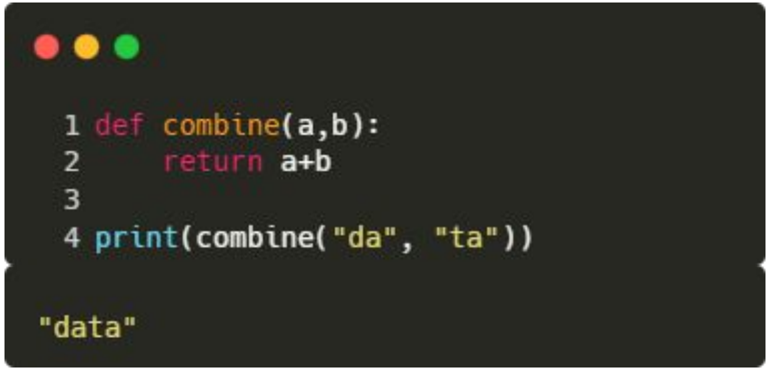
- Functions are a predefined block of code.
- A function has a name, input parameters, and an optionally an output parameter.

Functions: Concept



- Functions are a predefined block of code.
- A function has a name, input parameters, and an optionally an output parameter.

Functions: Example

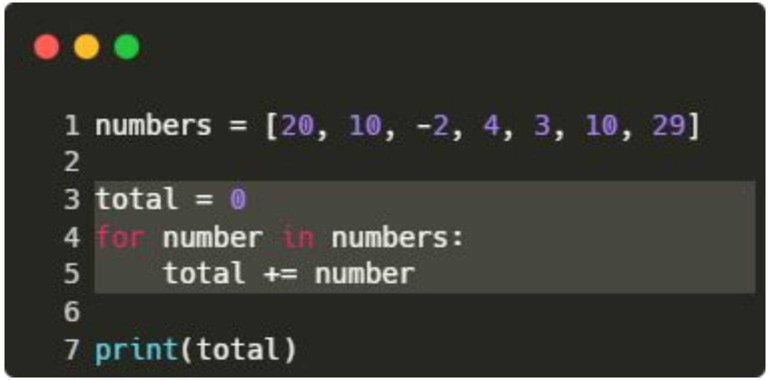


```
def combine(a,b):
    return a+b

print(combine("da", "ta"))
```

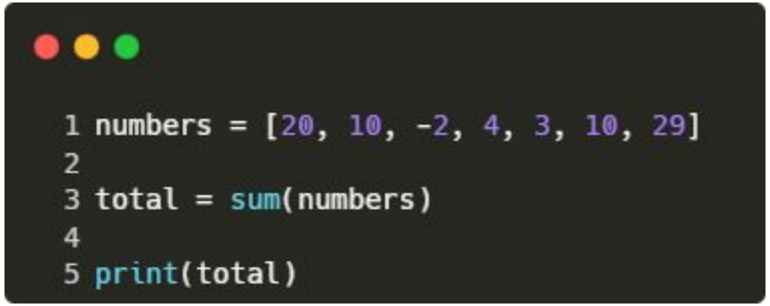
```
"data"
```

- Functions are a predefined block of code.
- A function has a name, input parameters, and an optionally an output parameter.

Functions: In-built



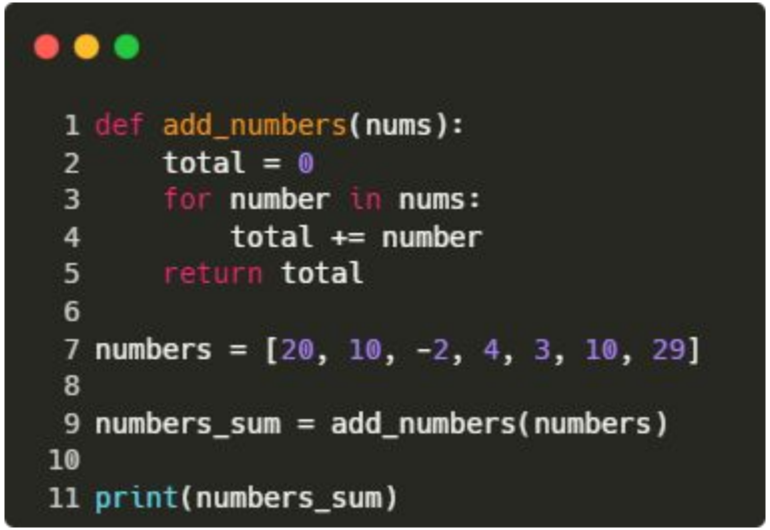
```
numbers = [20, 10, -2, 4, 3, 10, 29]

total = 0
for number in numbers:
    total += number

print(total)
```



```
numbers = [20, 10, -2, 4, 3, 10, 29]

total = sum(numbers)

print(total)
```

- Functions are a predefined block of code.
- A function has a name, input parameters, and an optionally an output parameter.
- Python has in-built functions like *sum()*, *print()*, *.append()*, etc.

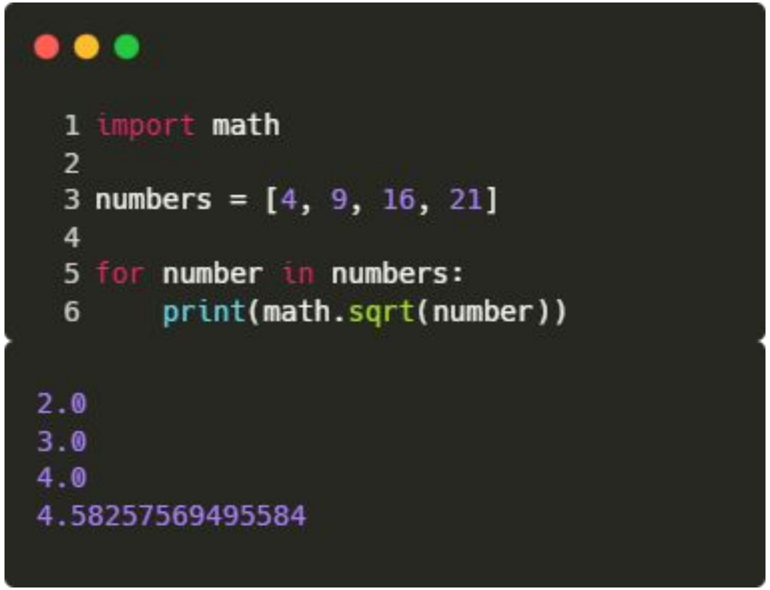
Functions: User-defined



```
def add_numbers(nums):
    total = 0
    for number in nums:
        total += number
    return total

numbers = [20, 10, -2, 4, 3, 10, 29]

numbers_sum = add_numbers(numbers)

print(numbers_sum)
```

- Functions are a predefined block of code.
- A function has a name, input parameters, and an optionally an output parameter.
- Python has in-built functions like *sum()*, *print()*, *.append()*, etc.
- You can write your own functions.

Functions: Libraries



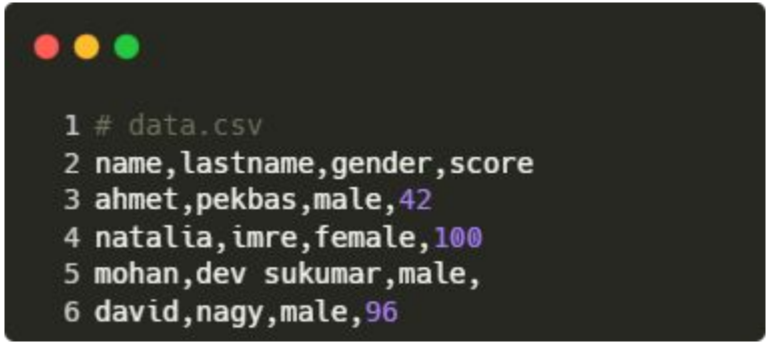
```
import math

numbers = [4, 9, 16, 21]

for number in numbers:
    print(math.sqrt(number))
```

```
2.0
3.0
4.0
4.58257569495584
```

- Functions are a predefined block of code.
- A function has a name, input parameters, and an optionally an output parameter.
- Python has in-built functions like *sum()*, *print()*, *.append()*, etc.
- You can write your own functions.
- Libraries are collections of functions which contain more functions.

Working with files

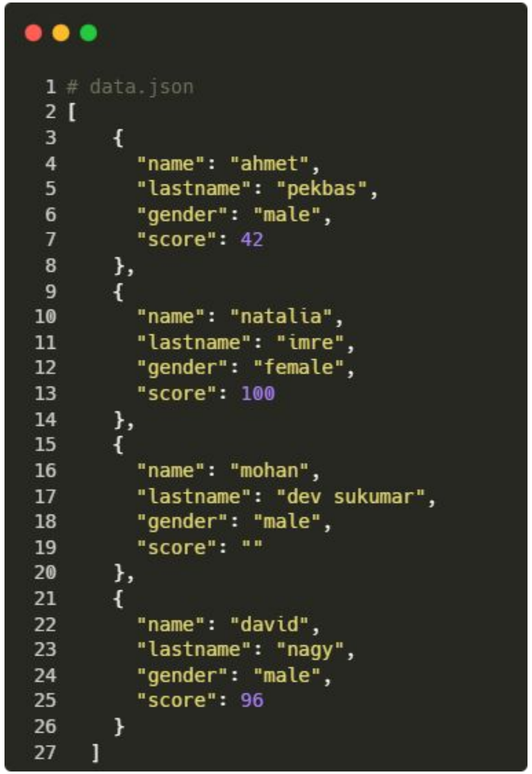
Working with Files



```
# data.csv
name,lastname,gender,score
ahmet,pekbas,male,42
natalia,imre,female,100
mohan,dev sukumar,male,
david,nagy,male,96
```

- We will work with **data** files.
- **.csv** and **.json** files are used to store data, but other file types exist.

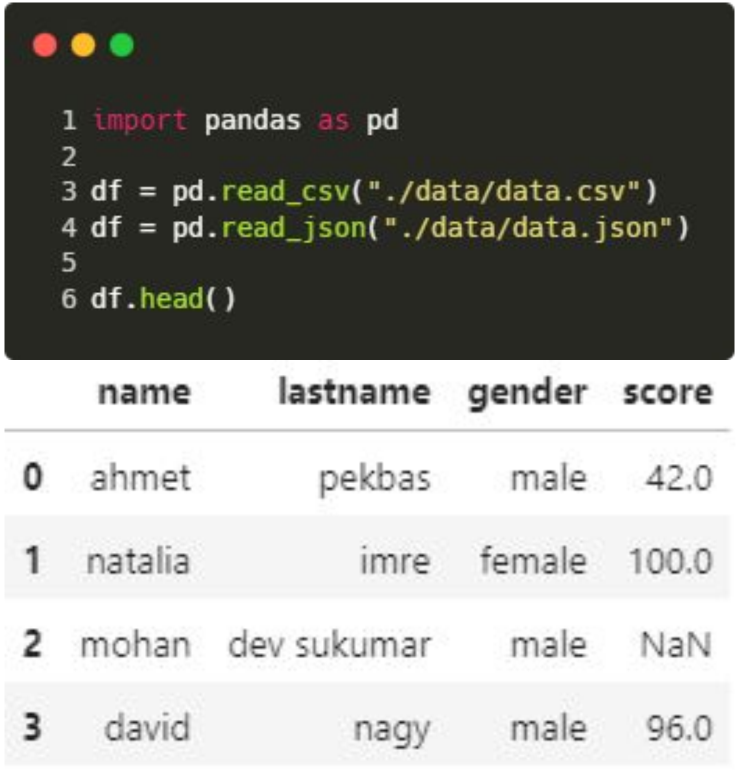
Working with Files



```
# data.json
[
    {
      "name": "ahmet",
      "lastname": "pekbas",
      "gender": "male",
      "score": 42
    },
    {
      "name": "natalia",
      "lastname": "imre",
      "gender": "female",
      "score": 100
    },
    {
      "name": "mohan",
      "lastname": "dev sukumar",
      "gender": "male",
      "score": ""
    },
    {
      "name": "david",
      "lastname": "nagy",
      "gender": "male",
      "score": 96
    }
  ]
```

- We will work with **data** files.
- **.csv** and **.json** files are used to store data, but other file types exist.

Working with Files



```python
import pandas as pd

df = pd.read_csv("./data/data.csv")
df = pd.read_json("./data/data.json")

df.head()
```

```
     name     lastname  gender  score
0   ahmet       pekbas    male   42.0
1 natalia         imre  female  100.0
2   mohan  dev sukumar    male    NaN
3   david         nagy    male   96.0
```

- We will work with **data** files.
- **.csv** and **.json** files are used to store data, but other file types exist.
- Pandas is a library to load, view, and edit these files.

Break - Then it’s your turn!