Data Structures & Programmatic Thinking

Pepe García jgarciah@faculty.ie.edu

2020-09-08

Plan for this session

Plan for this session

- Python basic datatypes

Plan for this session

- Python basic datatypes
- Variables

Plan for this session

- Python basic datatypes
- Variables
- Operators

Plan for this session

- Python basic datatypes
- Variables
- Operators
- Basic functions

Datatypes

Datatypes tell Python how we want to use the data. There are several primitive data types in Python such as **bool**, **int**, **str**, **float**.

Datatypes

Integers

Integers (or ints) represent whole numbers. We create them by using their numeric representation directly.

```
1
234
432432
```

Datatypes

Integers

Integers (or ints) represent whole numbers. We create them by using their numeric representation directly.

```
1
234
432432
```

Demo

Datatypes

Floating point numbers

Floats represent numbers that have a fractional part. We use a dot to separate the integer and fractional parts:

```
3.14
1.0
33.33
```

Datatypes

Floating point numbers

Floats represent numbers that have a fractional part. We use a dot to separate the integer and fractional parts:

```
3.14
1.0
33.33
```

Demo

Datatypes

Strings

Strings are used for textual representation. They can be created using either double or simple quotes.

```
'this is a string'
"this is another string"
```

Datatypes

Strings

Strings are used for textual representation. They can be created using either double or simple quotes.

```
'this is a string'
"this is another string"
```

Demo

Why can one use either double or single quotes? why just not agree on one of them?

Datatypes

Booleans

Booleans represent truthiness. There are only two values in for the bool type in Python: True and False

```
True
False
```

Datatypes

Booleans

Booleans represent truthiness. There are only two values in for the bool type in Python: True and False

```
True
False
```

Demo

Getting the type of a value

We can always get the type of a value using the **type(value)** function

```
type("patata")
```

Getting the type of a value

Practice

Inside Spyder, check what's the type of the following expressions:

- `"there is some text here"`
- `1`
- `True`
- `44.4`
- `'true'`
- `'False'`
- `2`
- `'33.3'`

Operators

Operators are symbols in the language that perform different kinds of computations on values

They're **binary**, they will operate on two values.

Arithmetic Operators

symbol	meaning
`+`	sum
`-`	substraction
`*`	multiplication
`/`	division
`**`	exponentiation
`//`	floored division
`%`	modulus

Arithmetic Operators

Rules of precedence

- Parentheses

Arithmetic Operators

Rules of precedence

- Parentheses
- Exponentiation

Arithmetic Operators

Rules of precedence

- Parentheses
- Exponentiation
- Multiplication/Division

Arithmetic Operators

Rules of precedence

- Parentheses
- Exponentiation
- Multiplication/Division
- Sum/Substraction

Arithmetic Operators

Rules of precedence

- Parentheses
- Exponentiation
- Multiplication/Division
- Sum/Substraction
- when operators have the same precedence, evaluate left to right

Arithmetic Operators

Rules of precedence

- Parentheses
- Exponentiation
- Multiplication/Division
- Sum/Substraction
- when operators have the same precedence, evaluate left to right

Arithmetic Operators

Rules of precedence

- Parentheses
- Exponentiation
- Multiplication/Division
- Sum/Substraction
- when operators have the same precedence, evaluate left to right

Demo

String operators

Sum and multiplication operators work on strings too. They're used to concatenate and multiply strings, respectively.

String operators

Sum and multiplication operators work on strings too. They're used to concatenate and multiply strings, respectively.

Demo

Variables

Variables are names that point to values in Python. We declare them using the assignment operator (`=`).

```python
variable_name = "value"
```

Variables

Naming variables

It's important to be as descriptive as possible when naming variables
There are some naming rules we should obey

Variables

Naming variables

It's important to be as descriptive as possible when naming variables
There are some naming rules we should obey

Rules

- variable names can't start with a number
- variable names can't contain special characters such as **`!`**, **`@`**, **`.`**
- Can't be one of the reserved words

Variables

Reserved words

`and`	`del`	`from`	`None`	`True`
`as`	`elif`	`global`	`nonlocal`	`try`
`assert`	`else`	`if`	`not`	`while`
`break`	`except`	`import`	`or`	`with`
`class`	`False`	`in`	`pass`	`yield`
`continue`	`finally`	`is`	`raise`	
`def`	`for`	`lambda`	`return`	

Variables

Mutability

In Python variables are mutable. This means that we can change their value at any time

```
name = "Pepe"
print(name)


name = "Jose"
print(name)
```

Converting values

There are some times when we need to convert a value from one type to another.

We use the **int()**, **bool()**, **str()**, and **float()** functions for that

Converting values

There are some times when we need to convert a value from one type to another.

We use the **int()**, **bool()**, **str()**, and **float()** functions for that

```
int('23')
bool(1)
bool(0)
str(True)
float("3.2")
```

Printing output

One can print output using the **print()** function

User input

There is a handy function **input()** that allows us to capture input from the user

```
name = input("Tell me your name: ")


print("hello, " + name)
```

Recap

- Datatypes (int, float, bool, str)

Recap

- Datatypes (int, float, bool, str)
- Variables (naming, mutability)

Recap

- Datatypes (int, float, bool, str)
- Variables (naming, mutability)
- Operators (arithmetic, precedence, string operators)

Recap

- Datatypes (int, float, bool, str)
- Variables (naming, mutability)
- Operators (arithmetic, precedence, string operators)
- Converting values

Recap

- Datatypes (int, float, bool, str)
- Variables (naming, mutability)
- Operators (arithmetic, precedence, string operators)
- Converting values
- User input