Data Structures & Programmatic Thinking

Pepe García

2020-04-20

Data Structures & Programmatic Thinking

Plan for today

- Python basic datatypes
- variables
- operators

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.

Integers

Integers (or ints) represent whole numbers without decimal parts. We create them by using their numeric representation directly

Integers

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
```

Floating point numbers

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"
```

Strings

Booleans

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

```
True
False
```

Booleans

Getting the type of a value

We can always ask ask a value for its type using the **type(value)** function

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

Getting the type of a value

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

Arithmetic Operators

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

Arithmetic Operators

Rules of precedence

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

Rules of precedence

String operators

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

String operators

Show some examples of string concatenation & multiplication

Variables

Variables are names that point to values in Python.

Variables

Naming variables

It's important to be as descriptive as possible when naming variables

There are some naming rules we should obey

Naming variables

Naming 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

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	

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)
```

Mutability

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")
```

Converting values

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)
- Variables (naming, mutability)
- Operators (arithmetic, precedence, string operators)
- Converting values
- User input

Exercises

1. Create a program that calculates the total number of seconds in an hour
2. How does the following expression evaluate?

   ```
   2 + (3 + 4) + (5 * 33 ** 34)
   ```

3. Create a program that asks the user for their age and their mother's age and calculate the age difference

4. What are the results and result types of the following expressions? think it yourself, do not use the Python console for this

   ```
   3 * 5 * 2
   3 / 11
   3 // 11
   25 % 2
   ```