Getting Started with Python

Let us hear some more about Python

1.

Data Types

See the Python basic data types in action

Values (Literals) and Data Types

- Examples of values in Python:
  - **1.23, 2000**,
  - **'Hello, world!'**,
  - **True**,
  - and more
- The data type determines the structure of an object
  - Which values are allowed?
  - What operations does the object support? For example, does it have a **`length`**? Can I add the object to another object?

Objects in Python

- Representation of the data
- Every object in Python has a **data type**, a value, and an identity

Objects in Python

- Representation of the data
- Every object in Python has a **data type**, a value, and an identity

Objects in Python

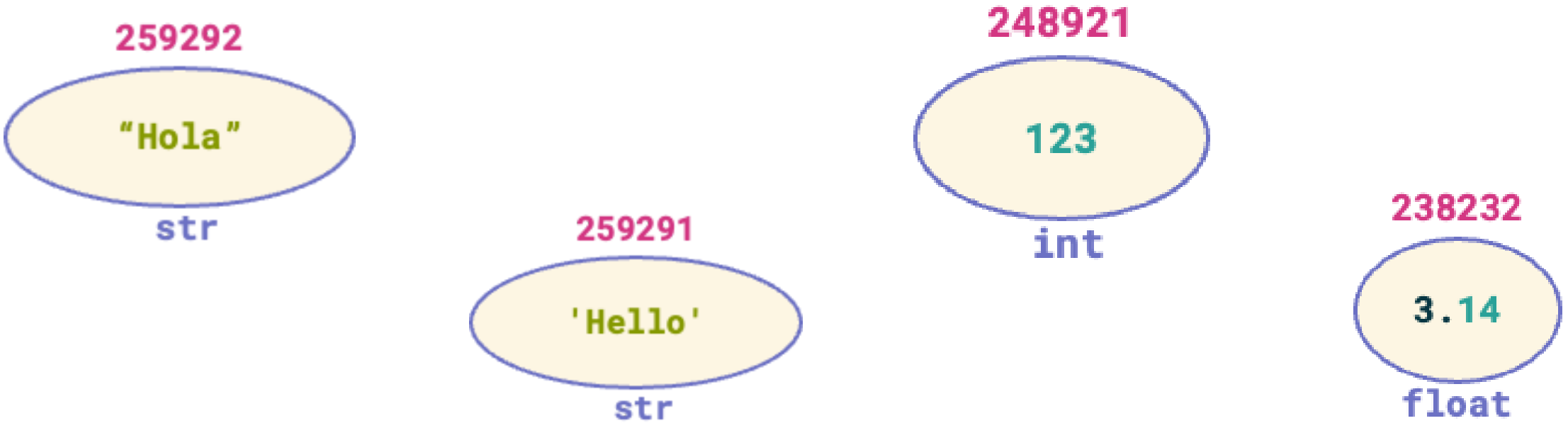
- Representation of the data
- Every object in Python has a **data type**, a value, and an identity

Objects in Python

- Representation of the data
- Every object in Python has a **data type**, a value, and an identity

Objects in Python

- Representation of the data
- Every object in Python has a **data type**, a value, and an identity



Data Types - Numbers

- Data type **`int`**
  - Represents integers
  - Example values: 2, 3, -100, 2000

- Data type **`float`**
  - Represents floating point numbers
  - Example values: 1.2, 3.14, 100.12, -13.23
    (note the **dot**)

Data Types - Strings

- Data type **str**
  - Represents textual data:

```
1 language = "Python"
2
3 course = 'Basics'
4
5 text = """Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla tempus rhoncus ultrices."""
```

Quiz-Time: Which Data Type?

Literal	Data Type
'Hello, there!'	?
'4123.314123'	?
3.14	?
9000	?

Arithmetic Operators

Precedence	Operator	Operation	Example
1	(...)	Brackets	(2 * 3) * 7
2	**	Exponent	2 ** 3
3	%	Modulo	10 % 2
4	/	Division	12 / 4
5	*	Multiplication	3 * 4
6	-	Subtraction	10 - 8
7	+	Addition	100 + 200

The Modulo Operator

- The modulo operator (**%**) is used to get the reminder of a division
  - The result of 15 divided by 4 is 12 with **remainder 3**

```
1 print(15 % 4) # Output: 3
2 print(17 % 12) # Output: 5
3 print(240 % 13) # Output: 6
```

Try it out: Arithmetic Operators

- Meaning of the operator depends on the data type of the operands.
  - Please create a new file (in your userfolder)

```
1 nr_1 = 4 * 6 + 4
2
3 text_1 = "Hello, " + "World"
4
5 text_2 = 3 * "Tatütata"
```

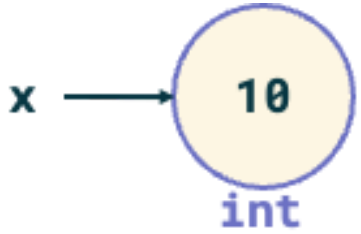
Quiz-Time: Operator Precedence

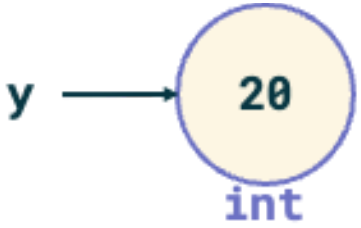
Expression	Result
2 ** 3 + 2 * 3	?
3 * (4 + 3)	?
2 ** (1 + 1) / 4	?

Variables

- A *variable* is a label that points to an object
- Value of the object can be accessed by means of variable names
- Declaration of a variable:

```
x = 10
y = 20
```





- Usage:

```
print(x + y)
```

Variables

- Follow rules for the definition of variable names
- Choose meaningful variable names
- Multiple assignments are possible

```
string1 = "David"
name = "David"
```

```
first_name, last_name = "Monty", "Python"
```

Variables - Reserved Keywords

- and, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while, yield

Variables

- We can print objects, variables, literals, etc. on the console using **`print():`**

```
1 print("Hello, world!")
2
3 nr_1= 10
4 nr_2 = 30
5 print("Result:", nr_1 + nr_2)
```

Quiz-Time: Output

What is the output when the following code is executed?

```
1 nr_1 = 10
2 nr_1 = 20.123
3 nr_1 = "Hallo"
4
5 print(nr_1)
```

Expressions

- An ***expression*** is a combination of **values**, **variables**, **function calls**, and **operators**
- **Important:** An expression is always evaluated to a value (literal)

```
20 + nr_1 + math.sqrt(4)
```

Statement

- A *statement* is an instruction that the Python interpreter can execute
- Examples for statements: assignment of variables, the definition of a function, **`while`** loop, **`for`** loop, etc.
- Multiple statements on one line must be separated with a semicolon (**;**)

```
name = 'David'
nr_1 = 10; nr_2 = 20
result = nr_1 + nr_2 + 40
```

Quiz-Time: Output

What is the value of variable **k** in the scripts **A** and **B** respectively?

A

```
x = 30; y = 20; z = 10
k = x + 10 - 4 * (y + z)
```

B

```
a = "a"; b = "b"
k = a + b + a + b * 3
```

Read User Input

Important: The input is always evaluated as **string**, no matter if the user enters a number

```
name = input("Your name: ")
print("Hello there,", name)
```

Example: Addition

Run the following code in PyCharm:

What is the problem with the following code?

```
nr_1 = 100
nr_2 = input("Enter a number: ")
print("Result =", nr_1 + nr_2)
```

Type Conversions (casting)

Conversion	**Code**
String → **Integer**	int('1234')
String → **Float**	float('3.14')
Float, Integer → **String**	str(3.14) *or* str(123)

Code Comments

- Comments are marked done a Hashtag (#)
- **Tip**: Comments should explain where necessary **why** certain instructions etc. are used

```
age = int(input("Age? "))
# Checks if age is greater than or equal to 18
# Only sell alcohol to people of legal age
if age >= 18:
  print("Enjoy your drink!")
else:
  print("Sorry, can't sell you a drink!")
```

Quiz Time: Variables - Reserved Keywords

- Which of them have we already used?

  and, **assert**, **break**, **class**, **continue**, **def**, **del**, **elif**, **else**, **except**, **exec**, **finally**, **for**, **from**, **global**, **if**, **import**, **in**, **is**, **lambda**, **not**, **or**, **pass**, **print**, **raise**, **return**, **try**, **while**, **yield**

Exercise

Write a script **simple_calculation.py** that implements the following statements:

1. **x** has the value **2**
2. **y** has the value **3**
3. **z** is the sum of **x** and **y**
4. Print the result of **(x * y) + z**

Exercise

Write a script **hello_there.py** that asks the user to enter his/her name and prints the following line by using the entered name:

```
Hello, <NAME>
```

<NAME> should be replaced by the entered name.

Exercise

Write a script **`simple_calculator.py`** that prompts the user to enter two numbers. The script should print the following string, where **<SUM>** is the sum of the two entered numbers:

```
The result is <SUM>
```

Recap

- Getting to know the Python interpreter and our setup
- Learn about objects, values, and data types
- Understanding statements and expressions
- Output text and read user input

End

That was all for this chapter