Information, Computation, Communication

Learning Python

Interactive Interpreter

Agenda

Interactive Interpreter: Launching

# To start interactive interpreter, type python in a terminal (console):

```
C:\> python
```

Careful: If you have both Python 2 and 3 installed on your PC, this command may launch Python 2. To launch Python 3, try typing python3 instead.

Interactive Interpreter: Launching

# To start interactive interpreter, type python in a terminal (console):

```
C:\> python
Python 3.10.12 (main, Jul 29 2024, 16:56:48)
[GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for
more information.
>>>
```

And right here you can start writing your Python code!

Interactive Interpreter: Printing Messages (1)

```
# Once interpreter is active, try writing this

>>> print("How do you do?")
```

Interactive Interpreter: Printing Messages (2)

```
# Once interpreter is active, try writing this

>>> print("How do you do?")
How do you do?
```

This is what you'll see being printed out in the terminal.

Strings (words inside quotes) get printed in the terminal

Interactive Interpreter: Computing (3)

```
# Let's introduce variables (symbols) and do some computation

>>> x = 2
```

Creating a variable whose name is x and value is 2
This variable is an integer

```
>>> x * 7
```

Let's multiply x by 7

```
14
```

It works!

Interactive Interpreter: Arithmetic and text? (1)

```
# Try replacing numbers with strings

>>> x = "Are you ready "
```

Besides numbers, variables can be strings

Interactive Interpreter: Arithmetic and text? (5)

# Try replacing numbers with strings

```
>>> x = "Are you ready "

>>> x + "to learn Python?"

'Are you ready to learn Python?'
```

Besides numbers, variables can be strings

What will happen if we use + (addition) with strings?

+ (addition) will combine (concatenate) two strings into one!

In Python, strings can be enclosed between apostrophes or double quotes

Interactive Interpreter: Asking for Input (1)

```
# Ask the user (well, the user is YOU now) to provide
# input to your program

>>> x = input("What value do you want x to have?")
```

Interactive Interpreter: Asking for Input (2)

```
# Ask the user (well, the user is YOU now) to provide
# input to your program

>>> x = input("What value do you want x to have?")

What value do you want x to have?
```

Your message to the user appears in the terminal and the program is waiting ...zzzzzzzz.... on you to answer!

Interactive Interpreter: Asking for Input (3)

# Ask the user (well, the user is YOU now) to provide
# input to your program

```
>>> x = input("What value do you want x to have?")
```

```
What value do you want x to have?
```

So let's answer.
For example...

```
77.045
```

Interactive Interpreter: Asking for Input (4)

```
# Ask the user (well, the user is YOU now) to provide
# input to your program

>>> x = input("What value do you want x to have?")

What value do you want x to have? 77.045

>>> print(x)
```

Did it work?
Let's print x to check

Interactive Interpreter: Asking for Input (6)

```
# Ask the user (well, the user is YOU now) to provide
# input to your program

>>> x = input("What value do you want x to have?")

What value do you want x to have? 77.045

>>> print(x)

 77.045
```

It worked!

Printing can be even simpler. Try: x
Result: '77.045'

Interactive Interpreter: Asking for Input (7)

```
# You can take text as input as well

>>> x = input("Say something…")

Say something…
```

Interactive Interpreter: Asking for Input (8)

```
# You can take text as input as well

>>> x = input("Say something…")

Say something…
```

Let's answer.
For example…

Interactive Interpreter: Asking for Input (9)

```
# You can take text as input as well

>>> x = input("Say something…")

Say something…Happy new semester!
```

What happened to x?
Let's print it to check

Interactive Interpreter: Asking for Input (10)

```
# You can take text as input as well

>>> x = input("Say something…")

Say something…Happy new semester!

>>> print(x)

Happy new semester!
```

Interactive Interpreter: Asking for Input (11)

```
# You can take text as input as well

>>> x = input("Say something…")

Say something…Happy new semester!

>>> print("You told me:", x)

You told me: Happy new semester!
```

Print can be used in many different ways!

Interactive Interpreter: Exiting

```
# To close interactive interpreter, type exit() in the terminal
# or press ctrl + z
```

```
>>> exit()
```

- Interactive interpreter is great, but…All we type is **temporary** and lost once the interpreter is killed
- **Better alternative**: write code in a file (known as scripting)

Next topic:
Numbers, operators, Booleans