HSLU Lucerne University of Applied Sciences and Arts

Python for Data Science: SW04

Python Debugging

Information Technology

March 12, 2025

FH Zentralschweiz

Content

Remote App Development

PyCharm vs. Jupyter Notebook

Debugging (in PyCharm)

- breakpoints
- start/pause/continue/stop
- online variable inspection

Enum data type

Ilias Exercises

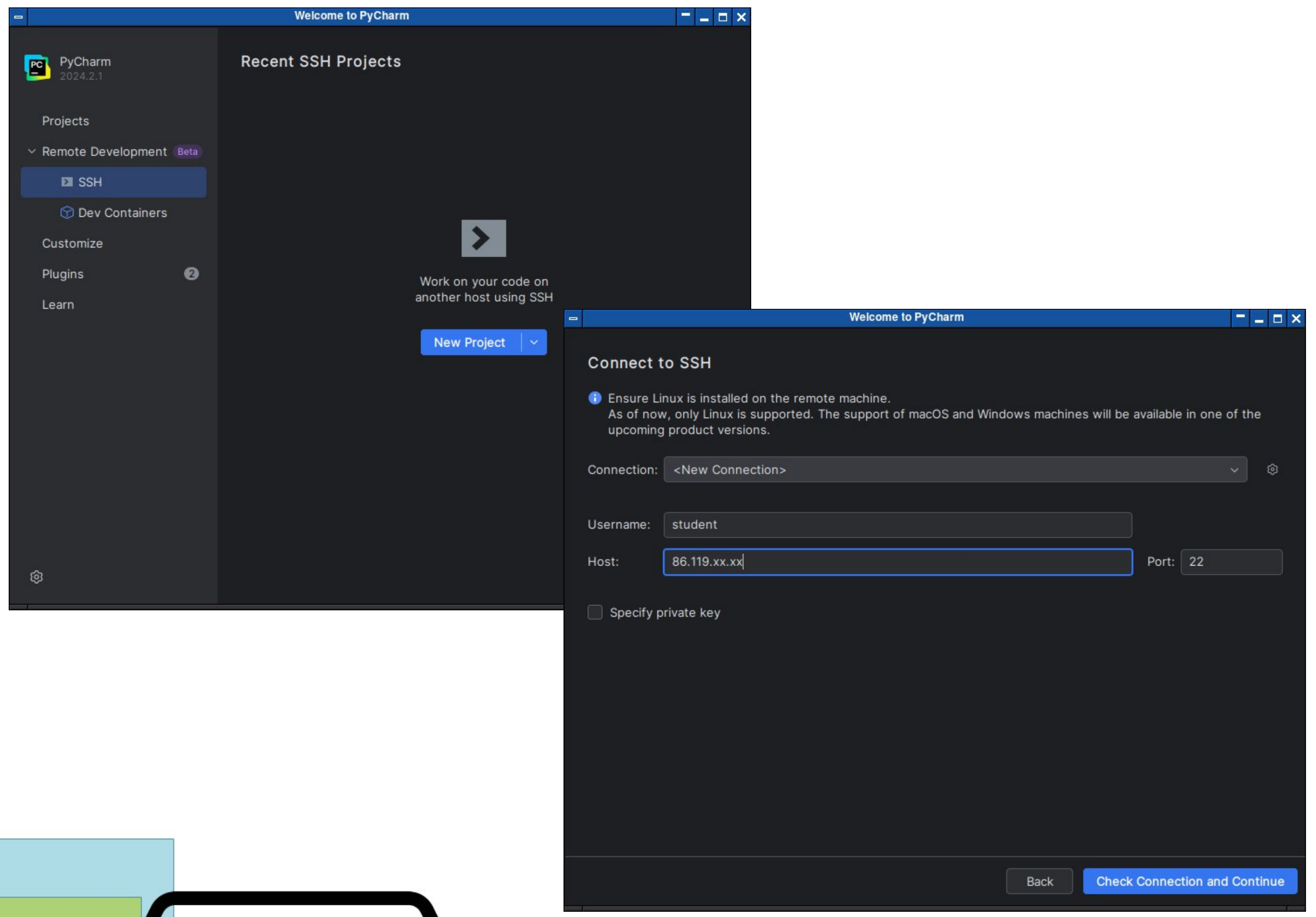
Remote App Development

Information Technology

March 12, 2025

Remote App Development

1. Connection to vm: start SSH project in PyCharm
2. Select Python interpreter in venv within vm (in this case, vm is local machine)
   -> if unable to click on OK, try to delete all venv folders and create new one.
3. Install plugin: PDF viewer



Work on your code on another host using SSH

Connect to SSH

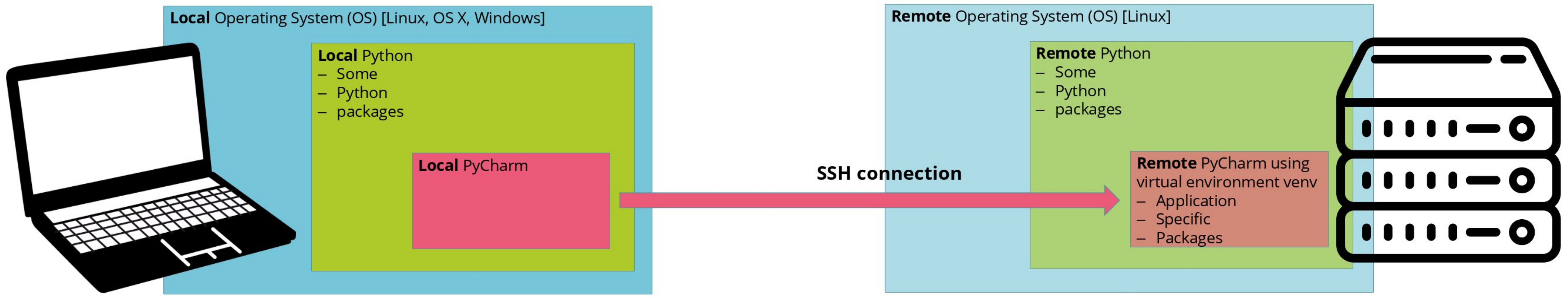
Ensure Linux is installed on the remote machine.
As of now, only Linux is supported. The support of macOS and Windows machines will be available in one of the upcoming product versions.

Connection: <New Connection>

Username: student

Host: 86.119.xx.xx Port: 22

Specify private key



Lucerne University of Applied Sciences and Arts

PyCharm vs. Jupyter Notebook

Information Technology

March 12, 2025

PyCharm vs. Jupyter Notebook

Python is the tool – but what is the development environment?

- Data scientists use two popular Python development environments for data analysis and reporting:

PyCharm

- Desktop IDE (from JetBrains)
- Community and professional edition
- Includes features like: code analysis, quick fix, auto-code generation, unit testing, etc.
- Full-fledged IDE with powerful debugger
- Console, terminal and file browser windows
- Used for complex data science projects

Jupyter (Julia, Python, R)

- Web application (server-client)
- Free software
- Provides different applications:
  Notebook, Lab (incl. basic debugger since 2020)
- Text and Code sections with PDF export
- In-line graphing
- Used for simple (often short) data science projects with focus on documentation

Debugging (in PyCharm)

Information Technology

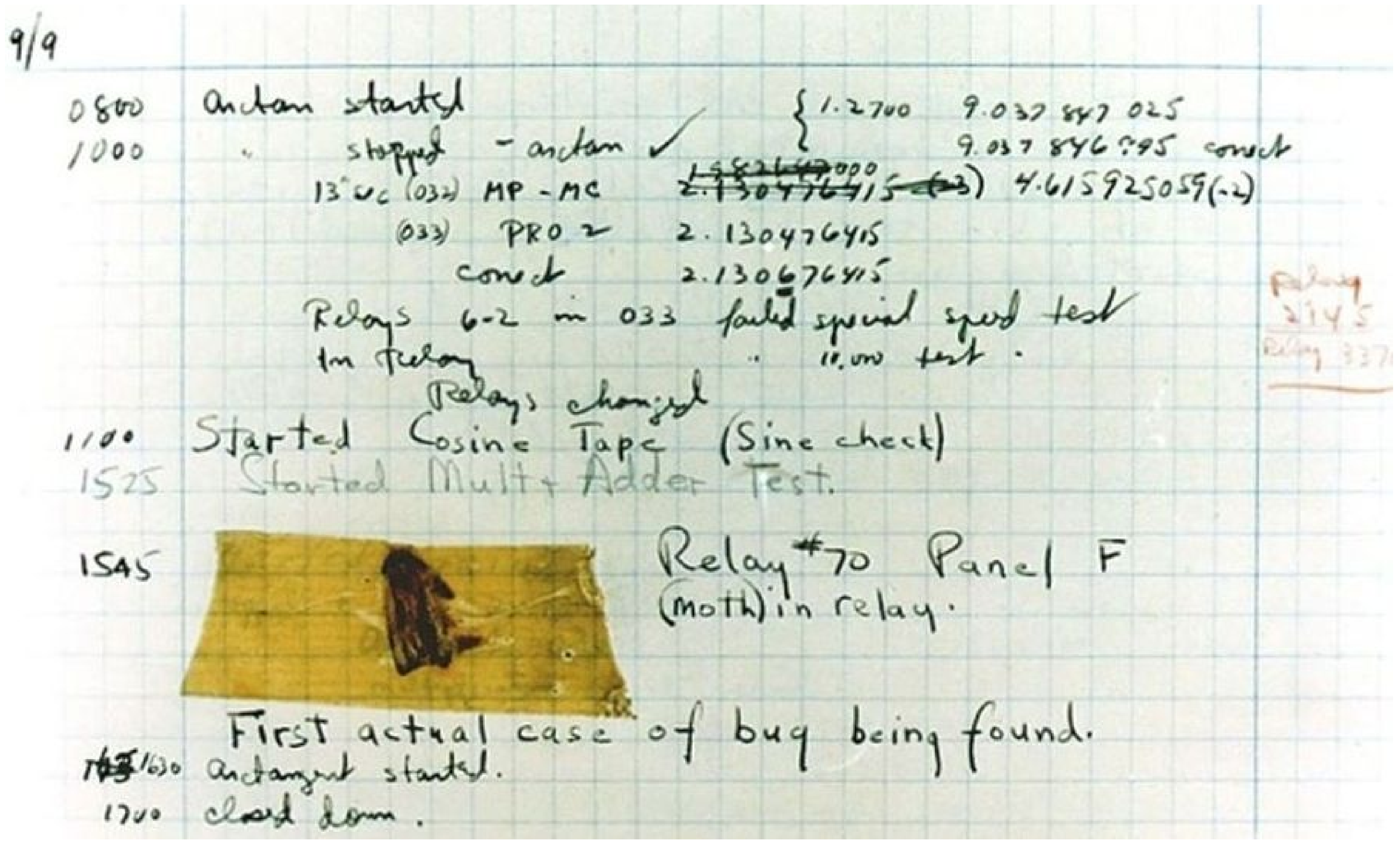
March 12, 2025

Debugging (in PyCharm)

Type of errors

- **Semantic errors**
  > violating rules of coding language.
- **Syntax errors**
  > missing code elements (e.g. parathesis).
- **Logical errors**
  > correct syntax but incorrect directions causing undesired output.
- **Runtime errors**
  > error happens when application is running or starting up.

https://www.ibm.com/topics/debugging



9/9

0800 Anctan started {1.2700 9.037 847 025

1000 " stopped - anctan ✓ 9.037 846 995 const

13" uc (033) MP - MC ~~1.982147000~~ ~~2.130476415~~ (-3) 4.615925059(-2)

(033) PRO 2 2.130476415

const 2.130676415

Relays 6-2 in 033 failed special speed test

In relay " 11.000 test .

Relays changed

1100 Started Cosine Tape (Sine check)

1525 Started Mult + Adder Test.

1545 Relay #70 Panel F (moth) in relay.

First actual case of bug being found.

1630 Antangent started.

1700 closed down.

Relay 2145

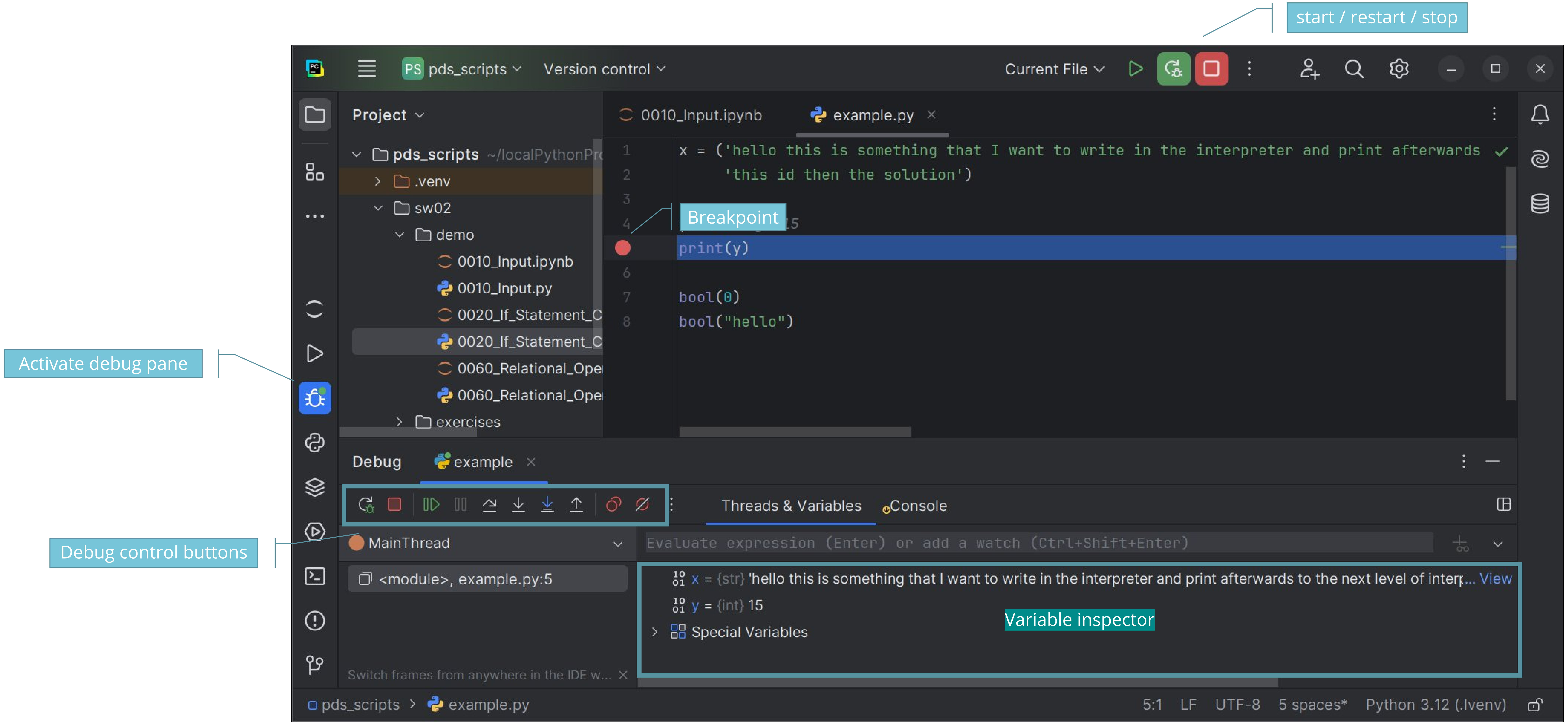
Relay 3370

In 1947 Grace Hopper and her staff started using the word “bug” to describe technical glitches when they found a dead moth (above) in Relay 70 of Harvard’s Mark II computer.Photo: NHHC Collection
https://spectrum.ieee.org/did-you-know-edison-coined-the-term-bug

Debugging typically involves six steps:

- Reproduce the conditions
- Find the bug
- Determine the root cause
- Fix the bug
- Test to validate the fix
- Document the process

Debugging (in PyCharm)



```
x = ('hello this is something that I want to write in the interpreter and print afterwards
     'this id then the solution')

                 5
print(y)

bool(0)
bool("hello")
```

```
Evaluate expression (Enter) or add a watch (Ctrl+Shift+Enter)
x = {str} 'hello this is something that I want to write in the interpreter and print afterwards to the next level of interp... View
y = {int} 15
> Special Variables
```

Debugging (in PyCharm): breakpoints

Breakpoints define code locations where the execution shall stop:

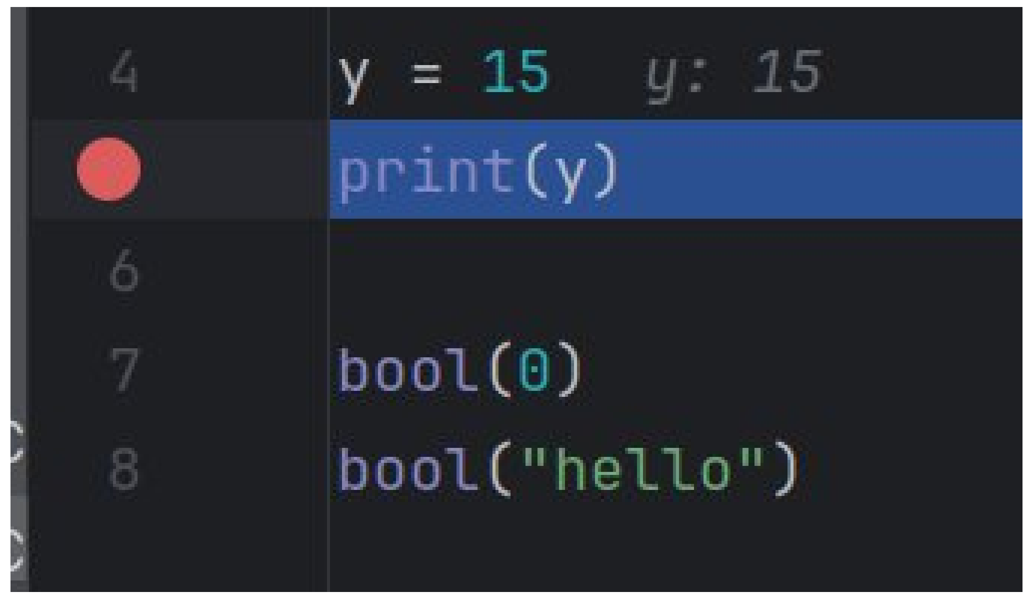
- a breakpoint is set/enabled by clicking on the line number.
  > widely used in IDEs.
- the execution is **stopped before** the selected line of code.
- breakpoints have no limits in amount.
- applicable at each instruction (code line).
  > unable to set for comments.
- can be disabled (not removed).

Regular breakpoints

- **always** stop execution.

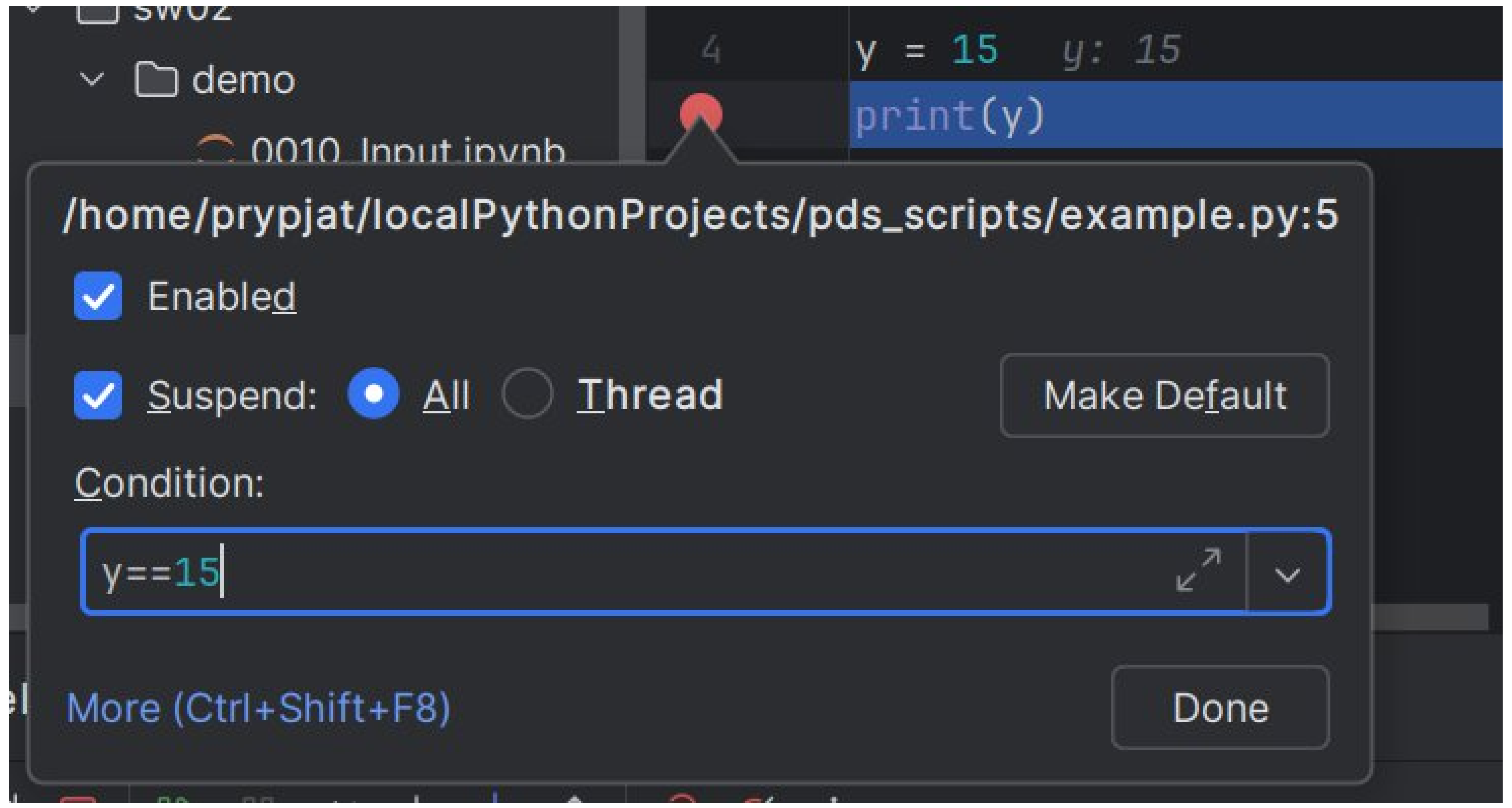
Conditional breakpoints

- only break if **condition** is **true**.
- can be "enabled" conditionally.



```
y = 15   y: 15
print(y)

bool(0)
bool("hello")
```



```
y = 15   y: 15
print(y)
```

/home/prypjat/localPythonProjects/pds_scripts/example.py:5

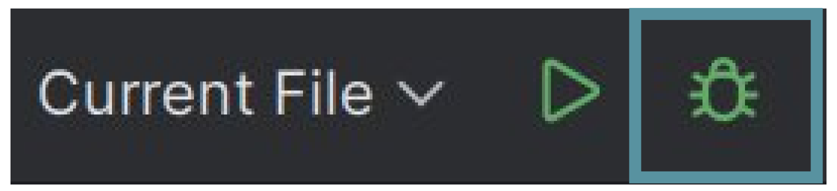
Enabled: ✓

Suspend: ✓ [✓ All] Thread

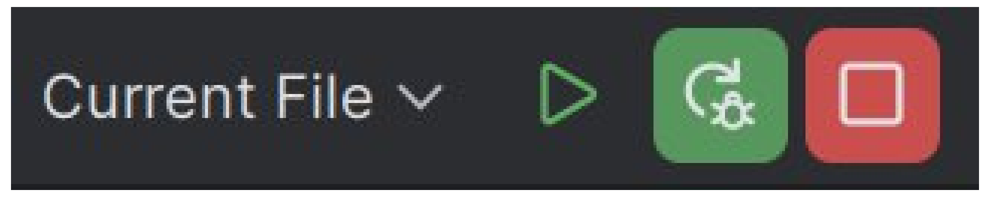
Condition: y==15

Debugging (in PyCharm): start/pause/continue/stop

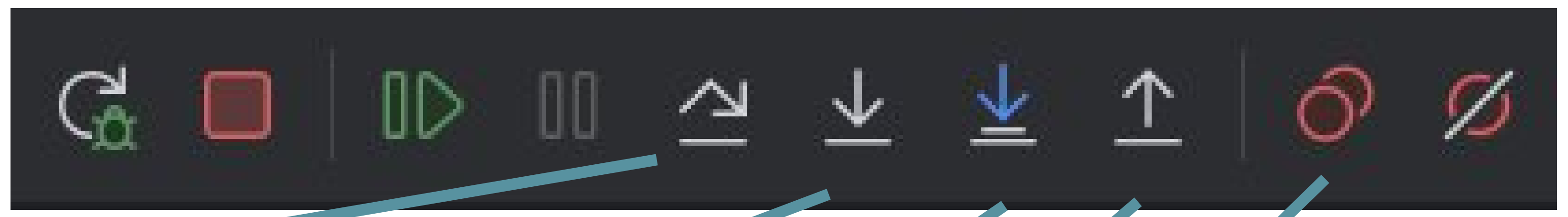
Start de-BUG:



Restart and Stop de-BUG:



Restart, Stop and Resume,



- Step over,
  > executing the full instruction without stopping.
- Step into
  > jump into instruction and stop at next code line.
- Step into my code
  > jump into instruction(s) if the code is part of the project.
- Step out
  > jump out of sub-instructions (instruction body)
- View break points

Debugging (in PyCharm): online variable inspection

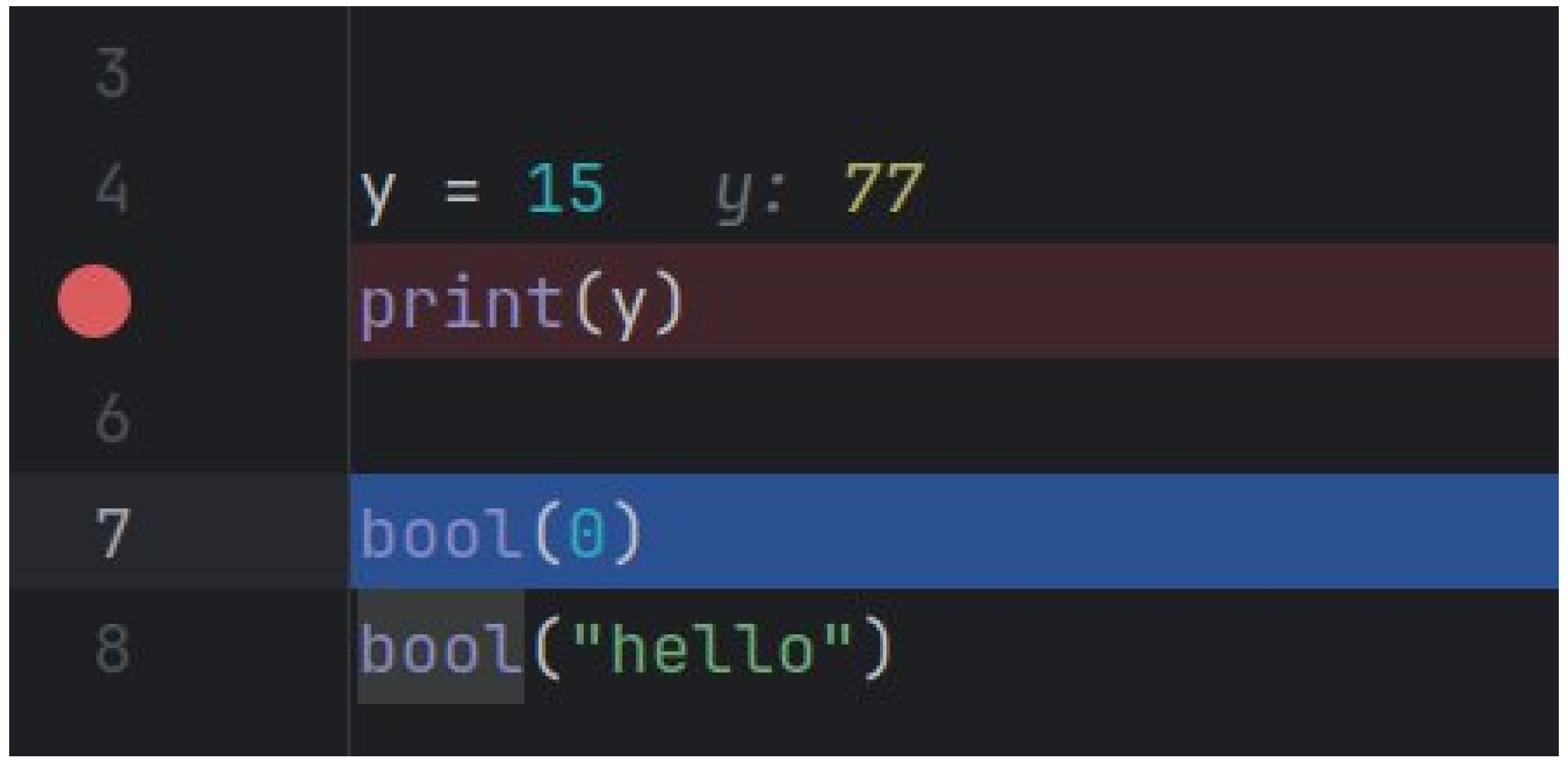
The variable inspector pane shows the current state of all active variables at the time of breaking execution. In modern IDEs, the current value of variables is typically presented next to each variable in the code.

The inspector allows to:

- inspect value,
- evaluate expression or
- edit current value.

E.g. the originally assigned value 15 of the variable y was changed to 77.

- force critical situation in order to provoke error.



```

y = 15   y: 77
print(y)

bool(0)
bool("hello")
```

Enum data type

Information Technology

March 12, 2025

Enum data type

An Enum is a set of **symbolic names** bound to unique values. They are similar to global variables, but they offer a more useful repr(), grouping, type-safety, and a few other features.

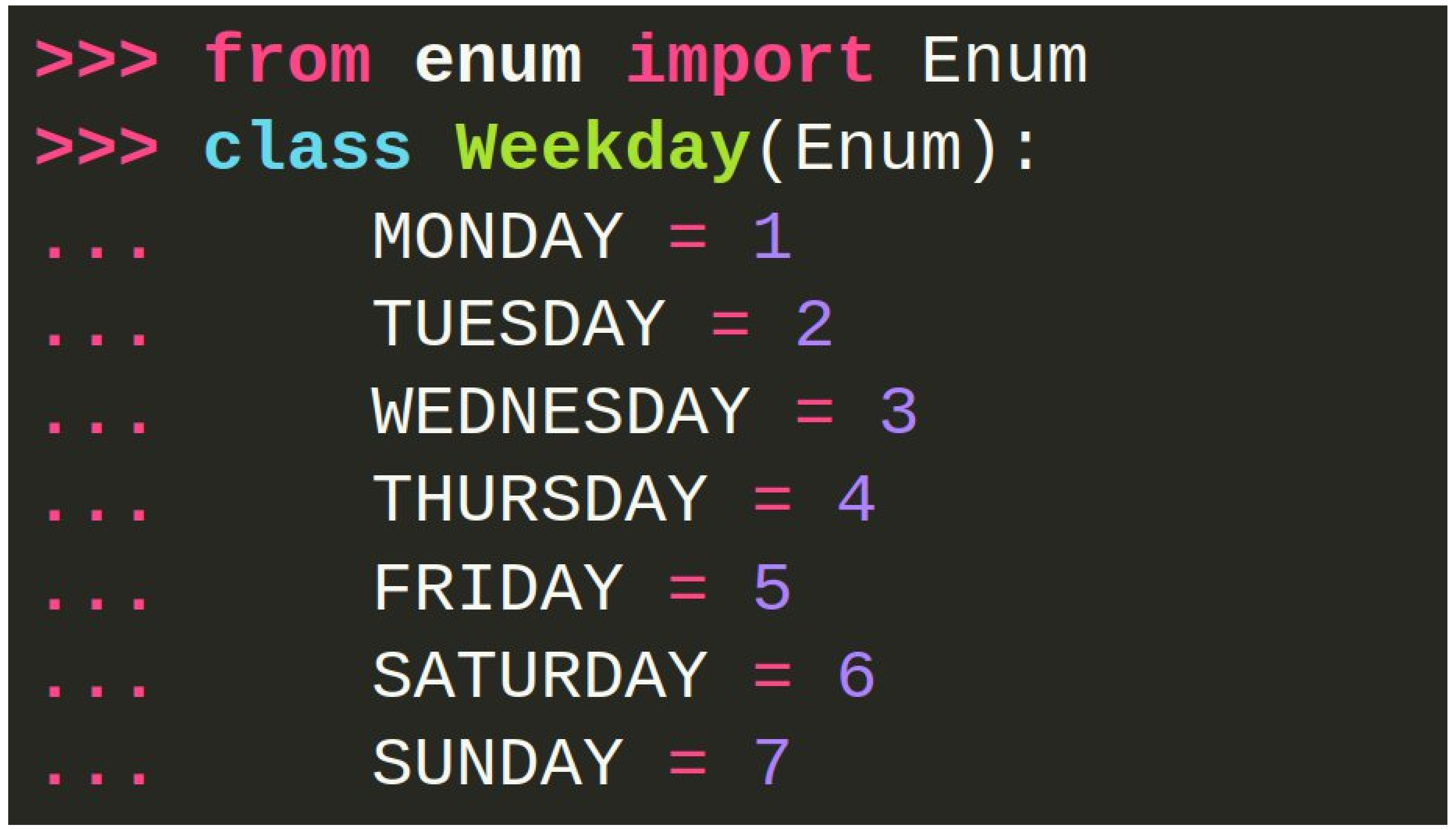
> Docu: https://docs.python.org/3/howto/enum.html

Enumeration requires the package **Enum:**

```
from enum import Enum
```

Enum allows handle a set of values:

- days of the week
- Sensor location: building level
- Type of apple
- Applicable form factors



```
>>> from enum import Enum
>>> class Weekday(Enum):
...     MONDAY = 1
...     TUESDAY = 2
...     WEDNESDAY = 3
...     THURSDAY = 4
...     FRIDAY = 5
...     SATURDAY = 6
...     SUNDAY = 7
```

Enum data type

Some key properties of Enum data type. Enum ...

... is similar to dictionaries having key – value pairs.

... members should be named in: `UPPERCASE`

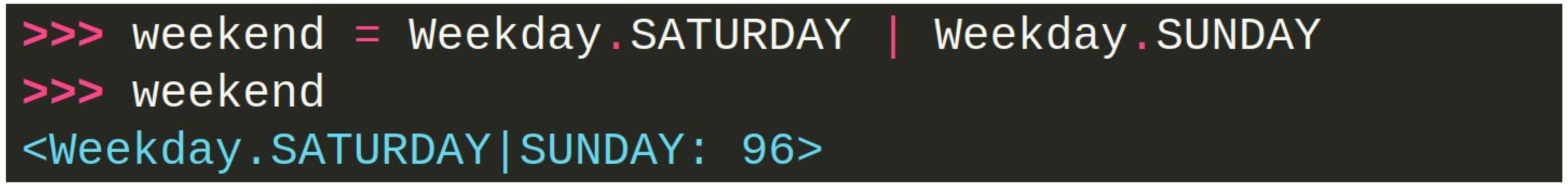
... is accessed by dot '.' notation: `y = Enumname.MEMBER`

... is used to represent constant.

... has no key functionality but provides more convenience for type handling.

... are not restricted to key – value pairs. **Python** allow Enum to provide functions.

In case of inheriting from **Flag** and assigning **values being power of 2**, Enum members can be combined:



```
>>> weekend = Weekday.SATURDAY | Weekday.SUNDAY
>>> weekend
<Weekday.SATURDAY|SUNDAY: 96>
```

HSLU Lucerne University of Applied Sciences and Arts

School of Computer Science and Information Technology
Research
Ramón Christen
Research Associate Doctoral Student

Phone direct +41 41 757 68 96
ramon.christen@hslu.ch

HSLU T&A, Competence Center Thermal Energy Storage
Research
Andreas Melillo
Lecturer

Phone direct +41 41 349 35 91
andreas.melillo@hslu.ch