Python course 1: Introduction to Python
Additional topics

Patrick Munroe

GERAD

2025-10-15

Type hints

Concept and syntax

- **Type hints** annotate variables and function signatures with types.
- **Syntax:**
  - Annotate variables:

    `variable: type`
  - Annotate functions:

    `def func(arg: type) -> returntype:`

  and
- **Motivation:** Improves code readability, helps with static type checking, and catches bugs early.
- **Examples:**

```python
# Annotate a function
def multiply(x: int, y: float) -> float:
    return x * y

# Annotate a variable
value: str = "hello"
```

Type hints

Example: greedy_solve

- Add type hints to the signature of our greedy_solve function:

```python
def greedy_solve(
        supply_by_sup: dict[str, float],
        demand_by_cust: dict[str, float],
        cost_by_sup_cust: dict[tuple[str, str], float],
        verbose: bool = False
        ) -> tuple[dict[tuple[str, str], float],
                   dict[str, float], dict[str, float]]:
```

- Add type hints to new variables, especially when the type is not obvious:

```python
flow_by_sup_cust: dict[tuple[str, str], float] = {}
```

Type hints

Example: `read_from_csv`

- Add type hints to the signature of our `read_from_csv` function:

```
def read_from_csv(
        file: str,
        index: str | list[str],
        column: str
        ) -> dict[str | tuple, float]:
```

- The operator `|` in `type1 | type2` indicates that the type can be either `type1` or `type2`.

List Comprehensions

Concept and syntax

- **List comprehensions** provide a concise way to create lists from iterables.
- **Syntax:**

  ```
  [expr(item) for item in iterable if condition]
  ```
- **Motivation:** Replace verbose for-loops with a single, readable line.
- **Examples:**

```
# List comprehension without a condition
squares = [x * x for x in range(10)]

# List comprehension with a condition
even_squares = [x * x for x in range(10) if x % 2 == 0]
```

List Comprehensions

Example: Create a flow matrix

- Convert the flows dictionary returned by `greedy_solve` into a matrix (rows: suppliers, columns: customers):

```
flows, rem_sup, rem_dem  = greedy_solve(supply, demand,
                                        costs)

# Create a matrix where each row is a supplier and each
# column is a customer
flows_matrix = [[flows[s, c] for c in demand.keys()]
                for s in supply.keys()]
```

- A "matrix" can be created by nesting two list comprehensions.

Dictionary Comprehensions

Concept and syntax

- **Dictionary comprehensions** provide a concise way to create dictionaries from iterables.
- **Syntax:**

  `{key_expr(item): value_expr(item) for item in iterable if condition}`
- **Motivation:** Replace verbose for-loops used to build dictionaries with a single, readable line.
- **Examples:**

```python
# Dictionary comprehension without a condition
squares_dict = {x: x * x for x in range(10)}

# Dictionary comprehension with a condition
even_squares_dict = {x: x * x for x in range(10) if x % 2 == 0
```

Dictionary Comprehensions

Example: Create a flow dictionary

- Extract a dictionary of flows from a model (for all supplier-customer pairs):

```
flows_opt = {(i, j): value(model.x[i, j])
             for i in model.S for j in model.C}
```

- Dictionary comprehensions allow you to build dictionaries efficiently, using a concise and readable syntax.

Heat Maps with `imshow`

- You can use `plt.imshow` to display a matrix as a **heat map**.
- **Example:** Visualizing the shipment flows:

```
flows_matrix = [[flows[s, c] for c in demand.keys()]
                for s in supply.keys()]

plt.imshow(flows_matrix)
plt.colorbar(label="Units shipped")
plt.title("Shipment Flows")
plt.xticks(ticks=range(len(demand)),
           labels=list(demand.keys()))
plt.yticks(ticks=range(len(supply)),
           labels=list(supply.keys()))
plt.show()
```

- `colorbar` adds a legend for flow values.
- `xticks`/`yticks` set axis tick positions and labels.