Type Annotations in Python

Terribly Intimidating or Tremendously Informative?

Gregory M. Kapfhammer

PyOhio 2021

```
def start(t: Talk) -> List[Fun, Learn]:
```

Okay, what is this about?

Key Questions

What are the **benefits** and **challenges** associated with using type annotations inside of Python program? Will types make me a better programmer?

Intended Audience

An **adventuresome** Python programmer who wants to explore how both a new **paradigm** and software **tools** can improve their development skills!

Let's explore type annotations in Python programs!

Python Program without Annotations

```
def extract_urls(df):
    """Extract a list of urls."""
    urls = []
    if "Url" in df.columns:
        urlc = df["Url"]
        if urlc is not None:
            urls = urlc.tolist()
    return urls
```

What is the type of `df`? The terrible docstring does not say!

What is the behavior of `return urls` in this function?

Python Program without Annotations

```
def extract_urls(df):
    """Extract a list of urls."""
    urls = []
    if "Url" in df.columns:
        urlc = df["Url"]
        if urlc is not None:
            urls = urlc.tolist()
    return urls
```

What happens if the program becomes more complex?

Python Program with Annotations

```
def extract_urls(df: pandas.DataFrame) -> List[str]:
    """Extract a list of urls."""
    urls = []
    if "Url" in df.columns:
        urlc = df["Url"]
        if urlc is not None:
            urls = urlc.tolist()
    return urls
```

What is the purpose of `df: pandas.DataFrame`?

Wait, isn't this more complicated?

Do type annotations have any benefits?

What are the trade-offs of type annotations?

Challenges

- *Readability* : function signatures are more difficult to read
- *Productivity* : programmers often must add type annotations
- *Complexity* : programs use many new classes and types

Benefits

- *Fail-fast* : quickly catch errors before running Python programs
- *Tooling* : text editors signal problems to programmers
- *Understanding* : developers understand the structure of data

Pyright language server in VS Code and Neovim

Mypy static type checker in terminal or editor

Easy command-line interface with Typer

Quickly find a defect that crashes a program

AnalyzeActions/WorkKnow

Command-Line Interface with Typer

```
 import typer
cli = typer.Typer()
@cli.command()
def download(
    repo_urls: List[str],
    repos_csv_file: Path = typer.Option(None),
    results_dir: Path = typer.Option(None),
    env_file: Path = typer.Option(None),
):
```

See `AnalyzeActions/WorkKnow` for details!

Command-Line Interface

```
Usage: workknow download [OPTIONS] REPO_URLS...
  Download the GitHub Action workflow history of repositories.
Arguments:
  REPO_URLS...  [required]
Options:
  --repos-csv-file PATH
  --results-dir PATH
  --env-file PATH
  --peek / --no-peek                      [default: False]
  --save / --no-save                      [default: False]
  --debug-level [DEBUG|INFO|WARNING|ERROR|CRITICAL]
                                          [default: ERROR]
  --help                                  Show this message and exit.
```

- Using type annotations, Typer can:
  - automatically generate all menus
  - perform error checking on all arguments
  - convert all arguments to the correct type

Defect Detection with Pyright

```
def create_results_zip_file(
    results_dir: Path, results_files: List[str]
) -> None:
    """Make a .zip file of all results."""
    with zipfile.ZipFile(
        "results/All-WorkKnow-Results.zip",
        "w",
    ) as results_zip_file:
        for results_file in results_files:
            results_zip_file.write(results_files)
```

Pyright Feedback in VS Code

```
Argument of type "List[str]" cannot be
assigned to parameter "filename" of
type "StrPath" in function "write"
```

```
with zipfile.ZipFile(
    "results/All-WorkKnow-Results.zip",
    "w",
) as results_zip_file:
    for results_file in results_files:
        results_zip_file.write(results_files)
```

`results_file`

Type Annotations in Python

Terribly Intimidating or Tremendously Informative?

Programmers define types

Automatically create command-line

Type checkers automatically find bugs

Type Annotations in Python

Yes, they are Tremendously Informative! Try them!

AnalyzeActions/WorkKnow

https://www.gregorykapfhammer.com/

gkapfham/pyohio2021-presentation