VERA C. RUBIN OBSERVATORY

Developer Topics

Kian-Tat Lim

NSF AURA U.S. DEPARTMENT OF ENERGY SLAC CHARLES AND LISA SIMONYI FUND FOR ARTS AND SCIENCES LSST CORPORATION

Python typing/mypy

- Middleware and SQuaRE teams are finding it helpful to add typing hints to Python code.

```
def append_pi(values: List[float]) -> None:
    values += [3.14]
```

- Satisfies Jim Bosch's inner C++ programmer
- Eliminates some bugs
- Helps indicate when interfaces are overly complex
- Will be integrated into `numpydoc` at some point (but not yet)
- Can run `mypy` to check via GitHub Action (see later), but requires `__init__.py` instead of namespace packages

pytest

- RFC-793 will allow pytest-only features to be used
- Can't run `python tests/test_foo.py` anymore, but there are simple alternatives
- Interesting pytest features — if they help:
  - parametrize to iterate over a list of test parameters
  - modular test fixtures to do things like set up temporary test directories or capture stdout/stderr
  - bare assert (but continue to use specialized `lsst` assertions for floats, images, etc.)

GitHub Actions

- Should be used today for linting Python, YAML, shell, and Markdown as well as mypy
- "Null" workflow to enable proper GitHub branch protection
- Note that templates are only used for creation; they do not stay updated
- Templates in `lsst/.github`; use "New workflow" button in Actions tab
- Can be used for other pre-merge tests
- Also used for building documents
- Limitations on available storage (14 GB) and multi-package builds mean that Jenkins is still around
- Almost all Travis usage should be moved to GHA; 41 remaining repos are mostly docs and obsolete dependencies

Adding A Workflow Using Templates

Choosing YAML vs. JSON vs. `pex_config`

- JSON for machines, YAML for humans
- If for machine-to-machine (program-to-program) communication, use JSON (much more efficient serializer)
- If for Web usage, use JSON
- If humans are going to edit or frequently read the file, do not use JSON
- `pex_config` is for pipeline Task configuration; expressed as Python in standalone files or in pipeline YAML files
- YAML tips:
  - `>` vs. `|` for multiline text (first merges into one line; second keeps line breaks)
  - `&` anchors and `*` references
  - `<<` merge key to include one dictionary in another

YAML Examples

```
- name: Python install
  run: |
    python -m pip install --upgrade pip
    python -m pip install "lander<2.0.0"
```

```
splenv_ref: &splenv_ref '0.6.0'
platform_defaults: &platform_defaults
  splenv_ref: *splenv_ref
platforms:
  - &el7-conda
    <<: *platform_defaults
    image: docker.io/lsstdm/scipipe-base:7
    label: centos-7-conda
    compiler: conda-system
    python: '3'
```

C++17 Features

- Standard library: filesystem, optional, any, string_view, polymorphic allocators, searchers, apply; parallelism including execution policies, reduce, inclusive_scan, exclusive_scan; mathematical special functions, std::gcd, std::lcm
- Removed `auto_ptr` (use `unique_ptr`)
- Simplified nested namespaces (`namespace lsst::afw::table { }`)
- Structured bindings for arrays, tuples, structs:

```
int a[2] = {1,2};
auto [x,y] = a; // creates e[2], copies a into e, then x refers to e[0], y refers to e[1]
```

- Initializers in `if` and `switch` statements:

```
if (auto it = m.find(10); it != m.end()) { return it->second.size(); }
if (char buf[10]; std::fgets(buf, 10, stdin)) { m[0] += buf; }
if (std::lock_guard lock(mx); shared_flag) { unsafe_ping(); shared_flag = false; }
```

Python 3.9/10 Preview

- Python 3.9 (maybe later this year, or skip?):
  - `dict` merge operators `|`, `|=`
  - Standard collections can be used directly in type hints
  - `str.removeprefix()`, `str.removesuffix()`
  - `zoneinfo` module for timezones
- Python 3.10 (maybe next year):
  - match/case with structured patterns (Nate Lust)
  - Parenthesized context managers (useful for multiple at once)
  - `zip` length checking
  - Typing: `Union[x, y]` as `x|y`; `TypeAlias` to define type variables
  - Deprecate `distutils`

rubin-env-extras

- `rubin-env` is the list of dependencies needed for production
- `rubin-env-extras` will include additional dependencies desired by users
  - Ensures that a compatible solution exists
  - Not necessarily something that anyone will install
- Additional subsets of `rubin-env-extras` beyond `rubin-env` may be defined
- Start thinking of packages that you'd like to use in conjunction with the Science Pipelines