ISO/IEC N 0405, Python Clarifications and Edits v00, Kevin Coyne

Issue #23

JWM	23	TH	E.45	TRJ-Python	The python annex equates TRJ (checking parameter values) with OTR (checking parameter types). This is an incorrect equation and the guidance given is appropriate.	Change E.45 to be similar to C.45 (the C annex).	

Response #23

I misunderstood the definition of TRJ and have corrected as below and have pattered the response after the C portion of the annex (D.45) as suggested.

E.45 Argument Passing to Library Functions [TRJ]

E.45.1 Applicability to language

Parameter passing in Python is called “passing by assignment” which, dependent on the mutability of the passed object, is analogous to passing by reference (when the object is mutable) or passing by value (when the object is immutable). In either case, the underlying mechanism is the passing of a pointer to the object.

Python supports all of the following argument passing capabilities:

- **Positional** - E.g., f(a,b,c)
- **Keyword** – E.g., f(a=1, b=10, z=99)
- **Keyword only** – Functions can specify which arguments must be specified as keywords.
- **Defaults** – Functions can specify values for parameters that are not passed.
- **Variable numbers of arguments** (a.k.a “varargs”). The variation can be in the number of arguments specified by the caller and/or the number specified by the function.

It is possible to pass invalid arguments to a function because it is not possible to statically check the types of arguments due to the dynamic typing capabilities of Python. However, Python catches any attempt to execute an operation that is not permissible for an object.

E.45. Guidance to language users

- Use the `type` function to check the type of passed objects if necessary to ensure that the passed object is of the correct type;
- Use default values for parameters when it will help guard against undefined parameters; and
- Do not make assumptions about the values of parameters.

Issue # 33

JWM	33	TL	E.28	XYQ-Python	The description doesn't add much to the LI description.	Replace first paragraph of E.28.1 with the following: "Python allows the usual sources of dead code (described in 6.28) that are common to most conventional programming languages." Insert a new first bullet in E.28.2 "Apply the guidance provided in 6.28.5."	

Response #33

Concur.

Issue #63

CA-28	63	TL	E.3.1		We dispute the statement that Python is strongly typed. By the common definitions of strong typing, such as strong guarantees about the runtime behaviour of a program, fixed and invariable typing of data objects, or the absence of unchecked run-time type errors, Python fails these tests and cannot be considered to be strongly typed.	Remove the statement.

Response #63

Python ***is*** strongly typed. Python, in fact, passes each of the aforementioned tests:

> Strong guarantees about the runtime behaviour of a program – It is not possible to perform an operation on a Python object that is inappropriate for its type. Further, the exception is caught at the point where the operation is attempted not at an obscure location which is possible, even commonplace, in weakly typed languages.
>
> Fixed and invariable typing of data objects – Objects are fixed in their type and cannot be changed. In fact, there is not even a way to cast objects to a different type – there is no concept of casting Python.
>
> Absence of unchecked run-time type errors – It is not possible to have runtime type errors – the interpreter will only perform operations that are permissible for the object's type.

There are additional explanations of why Python is a strongly typed language at: http://wiki.python.org/moin/Why%20is%20Python%20a%20dynamic%20language%20and%20also%20a%20strongly%20typed%20language

The fact that Python is also a ***dynamically***-typed language does not diminish its strongly typed character. There is some confusion about Python's strength in typing due to two unusual capabilities that were intentionally made part of the design of Python:

> **Variables do not have types** – only the objects that they point to do. Therefore, it is possible, and commonplace, to have a variable point to different types of objects at different times with no ill effects. This has no effect whatsoever on the types of operations that can be performed on the objects being pointed to.
>
> **Python does not support static typing.** I.e., there is no declaration of a variable's type (or more accurately in Python – there is no declaration of the type of the object pointed to by the variable). Typing is done dynamically but dynamic typing and strong typing are orthogonal – one does not preclude the other.

Issue #64

CA-29	64	E	E.8		Be consistent in language stating that the language does have a vulnerability.	Begin the clause with "This vulnerability is not applicable to Python since..."	

Response #64

Concur.

Issue #64

CA-29	64	E	E.8		Be consistent in language stating that the language does have a vulnerability.	Begin the clause with "This vulnerability is not applicable to Python since..."	

Response #64

Concur.