	Repo	Defect Name/ID/Link	Defect Description	Scenario Analysis	ODC	Cog Mode	Confidence
A1	Pytorch	https://github.com/pytorch/pytorch/pull/122618	Torch.export is used to share models or subgraphs in a way that allows results to be easily replicated. However, in the current implementation of export, the example_inputs field is thrown out. When trying to replicate bugs, benchmarks, or behaviors, losing the original input shapes and values makes the process much messier.	The "export" function should have easily verifiable outputs. But, the developers remove a needed field that would help the process.	Checking	HEM2	
A2	Pytorch	https://github.com/pytorch/pytorch/issues/118988	When a bound method is captured by a closure, we end up with a graph break. Calling a bound method locally traces fine. This is due to the DEREF of the outer variable being classified as UserObjectVariable(method). We should follow the same wrapping logic that we do for local vars.	The bound method breaks when not called locally. Developers need to use wrapping logic and treat the method as a local variable.	Checking	HEM1	
A3	Pytorch	check negative dilation by linlang1837 - Pull Request #122087 - pytorch/pytorch (github.com)	Check if dilation[0] is negative. This is an edge case that needs to be accounted for to avoid crashes.	The developers forgot to include an edge case that helps avoid crashes.	Checking	HEM6	
A4	Pytorch	Fix fuse linear bn for affine=False by gau-nernst - Pull Request #122537 - pytorch/pytorch (github.com)	"fuse_linear_bn_weights() needs to handle the case when batch norm weight and bias is None. Conv-BN version already handles that."	The method "fuse_linear_bn_weights()" does not handle the case that batch norm and bias are None.	Checking	HEM6	
A5	Pytorch	https://github.com/pytorch/pytorch/issues/121725	when setting dim=None, which according to the documentation should reduce across all dimensions, the function throws a RuntimeError instead:	Developers did not accurately code the outcome if the variable "dim" is set to None.	Algorithm	HEM2	
A6	pandas	BUG: Data type is float64	When a Dataframe is initialized with an array of shape (0,0), the data type is float64 regardless of the specified datatype of the array	Developers did not take into account the datatype of the initialized array when making a Dataframe.	Assignment	HEM5	
A7	pandas	BUG: When upsampling an empty dataframe	When upsampling an empty dataframe, the index type changes to DatetimeIndex from MultiIndex which it was supposed to be.	Developers did not keep index type consistent when upsampling dataframes.	Algorithm	HEM5	
A8	pandas	BUG: .rolling() returns incorrect values when ts.index is not nano seconds	The rolling() function returns incorrect values if the index is in micro or milliseconds-- only works with nano seconds.	Developers do not check the index type which results in incorrect values for everything but nanoseconds	Checking	HEM1	
A9	pandas	BUG: value_counts returning incorrect dtype for string dtype	`**SeriesGroupBy.value_counts**` returning incorrect dtype for string dtype. **self.obj.columns.dtype** needs to be specified (**dtype** = **self.obj.columns.dtype**)	Developers missing a parameter that needs be set for the output dtype to be correct.	Checking	HEM6	
A10	pandas	BUG: Empty list passed to Series returns object dtype, but via DataFrame returns float64	When an empty list is passed to a Series, the resulting **dtype** is **object**. However, when the list is passed to a DataFrame, the **dtype** is **float64**. The dtypes are supposed to be consistent between the constructors.	The pd.Series and pd.DataFrame constructors were probably implemented with different default behaviors for handling empty inputs.	Assignment	HEM1	
A11	numpy	UG: np.unique_inverse output has wrong shape	According to the specification, inverse indices should have the same shape as the input, but in the current implementation, the inverse indices is flattened.	The developer may have misunderstood or overlooked the specification that inverse indices should maintain the same shape as the input.	Algorithm	HEM2	
A12	numpy	Encoding an empty unicode array would produce an array of the wrong dtype	Calling numpy.char.encode on empty unicode array would create a float64 array instead of an array of S dtype. Expected behavior is an empty S array. It also incorrectly returns the shape:	The function numpy.char.encode might have a default behavior that returns a float64 array when dealing with empty inputs.	Assignment	HEM1	
A13	pandas	BUG: Float64Dtype series become numpy arrays with "object". dtype	Floating series causes failure when translating to a floating array. In **scipy.stats.entropy()**, **np.asarray(S)** outputs an array of object datatype when it was originally float64.	The developer did not properly code np.asarray(S) to be returned with the right output. It is an issue with the contents of the series.	Assignment	HEM2	
A14	numpy	BUG: Fix memory leak in timsort's buffer resizing	Memory leak was caused when resizing buffers due to a special case in the code that allocated memory for a NULL buffer.	The developer does not properly handle the case when the buffer is NULL, leading to incorrect memory allocation and eventual leaks.	Algorithm	HEM6	
A15	numpy	BUG: np.linalg.vector_norm with axis=None and keepdims=True returns the wrong shape	**np.linalg.vector_norm** with **axis=None** and **keepdims=True** returns the wrong shape: supposed to be (1, 1, 1) but code fails on assertion check.	Developers do not not correctly handle the keepdims=True parameter when axis=None.	Algorithm	HEM2	
A16	numpy	BUG: Fix ma.convolve if propagate_mask=False	**np.ma.convolve** raises a MaskError if **propagate_mask=False** and mode is set to 'same' or 'valid': fixes the issue by passing the mode keyword when creating the output mask. (The issue is fixed by....)	Developers have a mismatch in how the mask is handled within the np.ma.convolve function, specifically when propagate_mask=False. This is because the mode keyword wasn't included in the output mask.	Checking	HEM5	
A17	numpy	BUG: masked structured arrays cannot be compared to their unmasked counterparts	"While looking at the code, it seemed obvious that a correction to the mask shape done inside an if statement should really be outside."	Developers incorrectly placed the mask shape correction inside an if statement that it was not meant to be in.	Algorithm	HEM8	
A18	numpy	BUG: random.dirichlet(alpha) can return nans in some cases.	Specific instance: When all the values in alpha are less than 0.1, and alpha ends in two or more zeros, the components of the variates returned by dirichlet(alpha) corresponding to those final zeros will be nan.	Developers did not consider that small values of alpha can lead to numerical instability and precision errors during the computation.	Algorithm	HEM6	
A19	numpy	BUG: (1.25.0) ufunc.at returns wrong results when indices is an array containing negative ints	Numpy add function **numpy.add.at(x, indices, v)** should be equivalent to **x[indices] += v** but this is not the case when **indices** contains negative values.	The behavior of numpy.add.at() with negative indices was not coded to be consistent with the behavior of x[indices] += v.	Checking	HEM2	
A20	numpy	BUG: Fix the signature for np.array_api.take	The array_api take() doesn't flatten the array by default, so the axis argument must be provided for multidimensional arrays. However, it should be optional when the input array is 1-D, which the signature previously did not allow.	Developers incorrectly implemented the "take" function in the Array API regarding its handling of multidimensional arrays versus 1-D arrays.	Algorithm	HEM2	
A21	numpy	BUG: numpy 2.0 dividing np.uint8 by python int leads to OverflowError	In numpy 2.0, numpy integer types aren't compatible with python integer types for division. Leads to "OverflowError."	Developers did not account for changes introduced by Numpy 2.0 that affected the behavior of integer division.	Build/package/merge	HEM5	
A22	numpy	BUG: log1p output has less precision for the real part of small complex inputs	The real part of the output of numpy.log1p is rounded down to zero when the real part of a complex input is less than about 1e-15 . It is not supposed to round down.	Floating-point representation limitations caused an error	Algorithm	HEM2	
A23	numpy	piecewise returns array with same dtype as input, which may have undesired outcomes	It overrides the dtype that the function returns after processing the input.	The developer overlooked the intended behavior of the function regarding the returned datatype.	Algorithm	HEM1	
A24	numpy	BUG: floor_divide is not normal when calculating the input containing inf	According to IEEE754 standard, the result of division with infinity/negative infinity should be positive or negative infinity. This is incorrectly represented in the code, which results in "nan" instead of +/- infinity.	The developer did not align with IEEE 754 standard regarding the handling of division involving infinity and produces incorrect output.	Algorithm	HEM1	

A25	numpy	BUG: Error when formatting a failure in 'assert_array_almost_equal_nulp'	The function **numpy.testing.assert_array_almost_equal_nulp** has a formatting error in the code that generates the error message for a test failure when the data type is np.longdouble. The test should fail with an AssertionError that contains information about the failed comparison. Instead, we get a ValueError: Unsupported dtype float128.	The error message generation code does not properly handle the np.longdouble data type, an oversight from the developer.	Checking	HEM2
A26	numpy	Inconsistent failure to do numeric operations on object-typed masked arrays	Standard array numeric operations (such as .min and .max) error when done on masked arrays. (arrays that may have missing or invalid entries.) This is because the output of .min is an integer while it should be an ndarray object.	Developers caused inconsistencies in the return types of array operations when dealing with masked arrays.	Checking	HEM1
A27	pandas	BUG: Getting "Out of bounds on buffer access" error when .loc indexing a non-unique index with Int64 dtype	Indexing a DataFrame with an Int64 index will error with IndexError: Out of bounds on buffer access (axis 0) if the resulting dataframe has more than 10k values. This does not error when the index is a regular int64 dtype. The code is expected to return the indexed dataframe.	The developer does not properly handle indexing with the Int64 index data type. This is because the size of the dataframe is too large to handle these values.	Assignment	HEM6
A28	pandas	BUG: Assignment of pyarrow arrays yields unexpected dtypes	Creating arrays with pyarrow datatype does not keep the pyarrow datatype, instead changes datatype to Object. The pyarrow dt is supposed to be maintained.	Developers do not account for mismatch between the way PyArrow datatypes are handled and the expectations of the code or libraries used for array creation.	Assignment	HEM1
A29	pandas	BUG: groupby().groups with multiple group keys always treats sort as True	groupby().groups with multiple group keys always treats sort as True. The documentation states when sort=False, the groups will appear in the same order as they did in the original DataFrame. groupby() is sorting when not supposed to.	There is an oversight in the implementation of the groupby() function, causing it to always sort the groups regardless of the sort parameter.	Algorithm	HEM2
A30	pandas	BUG: Group keys contain NA values despite dropping them in groupby method	group_keys contains NA values even though all NA values were supposed to be removed.	Logic did not correctly cover all scenarios in removing NA values.	Algorithm	HEM6
A31	pandas	BUG: sqrt(ma.masked) returns non-masked array	In numpy 2.0, applying sqrt() to a fully masked scalar array undoes the mask. numpy 1.26.3 behaves correctly.	NumPy 2. introduced changes in the behavior of the sqrt() function, leading to unintended side effects with masked arrays.	Build/package/merge	HEM6
A32	numpy	BUG: average() returns nan when an inf value is given a 0 weight #	When computing the weighted mean of an array that includes infinity, and that element is given a weight of 0, code gives a RuntimeWarning, and a nan result.	The developer does not properly handle in the code the case where an element in the array is infinity and is given a weight of 0.	Algorithm	HEM2
A33	vscode	Error: Illegal value for lineNumber	When switching between empty files, "Error: Illegal value for lineNumber." This bug was fixed by checking that the effectiveLineNumber is valid before accessing line content.	The developer missed a parameter check.	Checking	HEM5
A34	vscode	Do not offer 'Create file' option for readonly file systems	Using Vscode.open, when on a readonly file system, trying to open a file that does not exist results in the option to "Create File."	The logic responsible for checking the existence of files may not be properly implemented or may not account for read-only file systems.	Interface	HEM1
A35	pandas	BUG: Inconsistent to_datetime behaviour	Incorrect datetime object generated when unix timestamp is passed as string. The timestamp string should be converted to the correct datetime regarless of the dtype of the input (string or int). Or, raise error if string inputs are illegal.	The code responsible for parsing the Unix timestamp from a string does not handle different data types (string or integer) correctly.	Algorithm	HEM1
A36	pandas	BUG: Series.pct_change raises error for empty series	Performing a pct_change on empty series raises error when fill_method=No_default, but works when fill_method=None. Expected behavior: Return empty result.	There may be inconsistencies in how the pct_change method handles different fill methods, leading to errors with certain options.	Algorithm	HEM2
A37	pandas	BUG: combine_first major performance regression	"In the reproducible example, the combine_first method takes 0.33s to run on my laptop with pandas 2.0.3, and 17s to runs on pandas 2.1.2"	Unintented changes occur between switching versions of pandas that the code is running.	Checking	Other
A38	pandas	BUG: DataFrame.from_dict loses bool type with NumPy MaskedArray	For a regular numpy array the bool type is preserved, but for a maskedarray it becomes object. Expected Behavior: Maintain boolean type	Developer incorrectly handles masked values	Checking	HEM2
A39	pandas	BUG: unexpected behavior in read_csv with pyarrow engine and dtype string	issue with read_csv when using the pyarrow engine: when a numeric column with missing values is imported as a string type, each number in the column has ".0" appended to it. occurs when setting the dtype to 'string' or 'string[pyarrow]', not observed if at least one of the data rows contains a string, or if the dtype is set to 'object'.	PyArrow's data type inference mechanism may interpret columns with missing values as floating-point numbers and coerce them into strings with ".0" appended.	Interface	HEM1
A40	Pytorch	Allow strided layout in torch.normal #111205	Strided layout doesn't work in Pytorch 2.1 while it used to work in Pytorch 2.0.	PyTorch 2.1 may have introduced changes in its memory management system that affect how data is laid out in memory, leading to issues with strided layout.	Build/package/merge	HEM1