Informed Consent

We are a group of researchers at the Technical University of Delft in the Netherlands. In this research project, we aim to investigate how people understand and respond to programming errors, with the goal of developing better ways of explaining such errors based on the user's programming proficiency. As such, you are invited to participate in our research study.

The following task is part of the research project described above. Upon accepting this consent form, you will first be asked to self-assess your experience with Python and general programming. You will then be presented with a series of multiple-choice questions focused on different aspects of Python programming errors. These questions are designed to help us objectively assess your Python proficiency. At the end, you will also be asked to estimate how many questions you believe you answered correctly. The entire task should take approximately 10 to 15 minutes to complete.

Completion of these tasks does not require any specific equipment. Your participation in this task is entirely voluntary, and you can withdraw at any time.

To ensure the integrity of our research, we kindly ask that you complete the task without the assistance of large language models (LLMs), AI tools, or external help. Your honest and independent responses are essential to the validity of our findings.

We will collect information related to your programming experience and how you interact with the survey. This includes:

- Programming experience: years of experience with Python and in general
- Interaction data: time spent on each question
- Responses from the questionnaire

We do not collect any data aside from the information described above, and we will keep your information confidential to the best of our ability. All data is stored in a password-protected electronic format. Be aware that the data we gather with this task might be published in anonymized form later. Such an anonymized data set would include the answers you provide in this task, but no personal information (e.g., your user name or ID), so that the answers will not be traceable back to you.

You can further contact the researchers for any clarification. To do this, send an email to amoraru@tudelft.nl for any questions.

By clicking the “**I consent**” option for this question, you confirm that you have read, understood, and consent to the above information.

Note: You can exit the task at any time. This will imply revoking your consent, and subsequently, all your data will be discarded from our databases.

- ○ **I consent**, begin the study
- ○ **I do not consent**, I do not wish to participate

Skill Level Self-reporting

How would you describe your experience in Python?

- ○ I have no previous experience in Python
- ○ Focusing on syntax & basics, relying on tutorials, lacking real-world project experience
- ○ Practical experience, small projects or assignments. Grasping basic concepts, and troubleshooting independently
- ○ Experience with projects. Able to independently plan and execute tasks, proficient with Python's libraries
- ○ Experience in complex projects. Deep understanding, ability to consciously resolve difficult problems
- ○ Significant experience in larger complex programs. Solves complex problems effortlessly and subconsciously

How many years of Python programming experience (YoE) do you have?

	0	5	10	15	20	25	30	35	40	45	50	
Number of YoE												

How many years of general programming experience (YoE) do you have?

	0	5	10	15	20	25	30	35	40	45	50	
Number of YoE												

Skill Level Assessment - General Programming Error Understanding

What is the main purpose of a programming error message?

- To notify the programmer that something is wrong in the code during compilation or execution
- To provide a warning about potential issues in the code
- To indicate that the code has been successfully compiled

- ○ To suggest optimizations for better performance

Which of the following statements best describes what a syntax error indicates?

- ○ The code does not conform to the rules of the language grammar
- ○ A file or resource cannot be found
- ○ The program logic is incorrect
- ○ There's an issue with variable naming conventions

What typically triggers a runtime error?

- ○ Trying to divide a number by zero
- ○ Missing semicolons at the end of statements (in programming languages that require them)
- ○ Using an undefined variable name
- ○ Improper indentation in Python

What are semantic/logical errors?

- ○ Errors due to syntax mistakes like missing parentheses
- ○ Errors detected by the compiler during code compilation
- ○ Errors where the program runs without errors but produces undesirable results
- ○ Errors related to file system permissions

What does a "type mismatch" error usually indicate?

- ○ A value is assigned or passed that doesn’t match the expected data type
- ○ A variable is being used before it's initialized
- ○ The function has too many return statements
- ○ The code is missing a necessary import or include

What distinguishes a compilation error from an interpretation error?

- ○ Compilation errors only occur in dynamically typed languages
- ○ Compilation errors occur before execution; interpretation errors may happen during execution
- ○ They are the same; the terms are interchangeable
- ○ Compilation errors occur at runtime, interpretation errors do not

Why might a program fail to compile even if the logic seems correct?

- ○ The program uses inefficient algorithms
- ○ The program takes too long to execute
- ○ The code violates the language's grammar rules
- ○ The output is not what was expected

Skill Level Assessment - Python-Specific General Error Understanding

Which error message typically indicates that a Python variable has not been defined?

- ◯ `SyntaxError: invalid syntax`
- ◯ `NameError: name 'x' is not defined`
- ◯ `IndentationError: expected an indented block`
- ◯ `TypeError: unsupported operand type(s)`

What does the following Python error message indicate?

```
TypeError: can only concatenate str (not "int")
to str
```

- ◯ A variable is used before it has been assigned a value
- ◯ The code tries to open a file that doesn't exist
- ◯ There's an issue with function call syntax
- ◯ Two incompatible types are being combined without explicit conversion

What does the following Python error message indicate?

```
IndexError: list index out of range
```

- ○ A syntax error related to list comprehension
- ○ The code does not have enough elements in a list to perform the operation
- ○ The code attempts to access an element outside the boundaries of a list
- ○ An invalid data type is being used in a string formatting operation

What does the following Python error message indicate?

```
AttributeError: 'int' object has no attribute
'append'
```

- ○ You're trying to use a method on the wrong object type
- ○ You're trying to index an integer like a list
- ○ You're trying to use a list method on a variable that holds an integer
- ○ You're calling a method that belongs to strings, but on an integer

What does the following Python error message indicate?

```
ZeroDivisionError: division by zero
```

- ○ The code is trying to divide zero by a number
- ○ The code is missing a return statement
- ○ The code is trying to concatenate incompatible types
- ○ The code is trying to divide a number by zero

What does the following Python error message indicate?

```
IndentationError: unexpected indent
```

- ○ A block of code is missing a required indentation
- ○ A block of code uses inconsistent indentation (spaces vs. tabs)
- ○ The code has an extra indentation where none was expected
- ○ A function was called before it was defined

What does the following Python error message indicate?

```
TypeError: 'int' object is not subscriptable
```

- ○ You're trying to loop through an integer as if it's a list or iterable
- ○ You’re trying to call an integer as if it were a function
- ○ You passed an integer to a method that expects a string
- ○ You tried to access an element of an integer using square brackets

Skill Level Assessment - Error Identification

Given the code snippet below, identify the line where an error occurs and why.

```
1: def greet(name):
2:     print("Hello, " + name)
3:
4: greet(John)
```

- Line 4: `John` is not defined (`NameError`)
- Line 2: Concatenation type mismatch
- Line 4: `John` is treated as a function but is not defined
- Line 1: Function definition error

The following code snippet is known to include an issue that will cause an error when executed. Choose the option that correctly identifies the type of error that will be produced.

```
x = "100a"
y = int(x)
```

- `NameError: name 'x' is not defined`
- `SyntaxError: invalid syntax`
- `ValueError: invalid literal for int() with base 10: '100a'`
- `TypeError: int() cannot convert letters to integers`

The following code snippet is known to include an issue that will cause an error when run. Identify the line and type of error that will be produced when the code is executed.

```
1: x = 10
2: if x > 5
3:     print("x is greater than 5")
```

- Line 2: `SyntaxError` due to extra comparison
- Line 2: `SyntaxError` due to missing colon
- Line 3: `IndentationError` due to wrong indent
- Line 1: `NameError` due to undefined variable

Given the following code snippet, what is the most likely error message one will receive when trying to execute it.

```
x = [1, 2, 3]
print(x[3])
```

- `TypeError: list index must be an integer`
- `NameError: name 'x' is not defined`
- `IndexError: list index out of range`
- `TypeError: 'list' object is not callable`

Read the code snippet below, and identify the error that is present.

```
data = {"key": "value"

print(data["key"])
```

- ◯ TypeError: 'dict' object is not callable
- ◯ SyntaxError: unexpected EOF while parsing
- ◯ KeyError: 'key'
- ◯ SyntaxError: missing colon in dictionary key-value pair

Given the following code snippet, what is the most likely error message one will receive when trying to execute it?

```
x = "5"

y = 3

print(x - y)
```

- ◯ IndexError: invalid index operation
- ◯ TypeError: cannot perform subtraction between string and number
- ◯ ValueError: cannot subtract string from int
- ◯ TypeError: unsupported operand type(s) for -: 'str' and 'int'

The following code snippet produces an error and does not execute correctly. Pick the answer that represents the most likely error.

```
def foo():
    print("Starting function")
    x = 5
    if x > 3:
        print("x is greater than 3")
      print("x is not greater than 3")
foo()
```

- ◯ IndentationError: expected an indented block after 'if' statement
- ◯ TypeError: unsupported operand type(s)
- ◯ NameError: x is not defined
- ◯ IndentationError: unindent does not match any outer indentation level

Skill Level Assessment - Error Resolution

The following code snippet is known to include an issue that will cause an error when executed. Choose the option that correctly resolves the error while maintaining the desired result.

```
# Print all non-negative integers until 5 included
```

```
for i in range(6)

    print(i)
```

- ○ Replace `range(6)` with `[0,5]`
- ○ Add a colon (i.e. `':'`) at the end of the `for` line
- ○ Add an `if` statement to check for non-negative integers
- ○ Remove the `for` keyword

The following code produces an error when executed. Which change would fix the error while preserving the intended behavior described in the docstrings?

```
def sum_list(lst):

    """

    This function computes the sum of a list of numbers.

    """

    total = 0

    for i in lst:

        total += i

    return total


# Sum of the list must be 10

print(sum_list([1, 2, "3", 4]))
```

- ○ Remove the for loop and use `sum(lst)` instead
- ○ Use a `try` block around the loop to skip the error
- ○ Remove the string `"3"` from the list

○ Convert each element to `int` before performing addition

The following code produces an error when executed. Which change would fix the error while preserving the intended behavior described in the docstrings?

```
def power(base, exponent):
    """
    This function computes the power of a number.
    If an exponent is not provided, we assume it to be 0.
    """
    return base ** exponent

print(power(2))
```

○ Change the operator to multiplication (i.e. '`*`') instead of exponentiation (i.e. '`**`')
○ Rename the function to avoid conflicts with built-ins
○ Remove the exponent parameter from the function definition
○ Provide a default value for the exponent

The `countdown` function below sometimes results in an error when called with certain inputs. Which change would fix the error while preserving the function's intended behavior, as described in the docstrings?

```
def countdown(n):

    """

    This function recursively counts down from n to 0.

    """

    if n == 0:

        print("Blast off!")

    else:

        print(n)

        countdown(n - 1)


countdown(-3)
```

- ○ Change the `else` statement to an `elif` statement
- ○ Replace recursion with an iterative `for` loop approach
- ○ Add a default value for `n`
- ○ Change the base case to `if n <= 0:`

The following Python code does not correctly print the double of a number. Choose the option that best resolves the logical error while maintaining the intended functionality.

```
double = lambda x: x * 2

print(double)
```

- ○ Surround `x * 2` with `print()`
- ○ Call the `double` function by adding parentheses with an argument
- ○ Replace `lambda` with `def` to make it work
- ○ Remove the `lambda` and just write `x * 2`

The `first_char` function below sometimes results in an error when called with certain inputs. Which change would fix the error while preserving the function's intended behavior, as described in the docstrings?

```
def first_char(s):
    """
    This function returns the first character of a string.
    If the string is empty or None, it returns None.
    """
    if len(s) > 0:
        return s[0]
    else:
        return None

print(first_char(None))
```

- ○ Convert `None` to an empty string before passing it to the function
- ○ Check if `s` is not `None` before calling `len()`
- ○ Always return `s[0]` without checking the length
- ○ Replace `None` with an empty string in the function

The code below is intended to apply a square function to each number within a list, but it raises an error. Choose the correct fix that maintains the logic.

```
nums = [1, 2, 3]

squared = map(lambda x: x**2, nums)

print(squared[0])
```

- ○ Use `list(squared)[0]` to access the first value
- ○ Replace map with a list comprehension
- ○ Use `squared = lambda x: x**2`
- ○ Change `lambda x: x**2` to `lambda x: pow(x, 2)`

Skill Level Assessment - Code Reading / Understanding

Consider the code snippet below. What option correctly explains the behavior of the following program?

```
x = int("0")

if x != 0 and (10 / x) > 2:

    print("Division succeeded")

print("Could not divide by " + str(x))
```

- The code contains a syntax error and won't run properly
- The code contains a `TypeError` as `"0"` can be ambiguously interpreted to different primitive types
- Division by zero error is avoided due to short-circuit logic
- The condition should use `or` instead of `and`, because `and` forces evaluation of both sides of the condition, which throws an error

Read the code snippet below and identify which code line is logically wrong when computing the area of a rectangle.

```
1: def area_rectangle(length, width):
2:     """Compute the area of a rectangle and print it."""
3:     area = length * width
4:     print("The area is", area)
5: area_rectangle(5, 10)
```

- Line 2 – Incorrect docstring format
- Line 3 – Incorrect multiplication operator
- Line 4 – Print statement formatting error
- No line; the code is logically correct

Read the code snippet below and identify what the expected output is when the code is executed.

```
def get_greeting():
    print("Hello, World!")
```

```
message = get_greeting()

print(message)
```

- ○ Two lines of `Hello, World!`
- ○ An error because the function returns nothing
- ○ `Hello, World!` followed by `None`
- ○ Only `Hello, World!`

Read the code snippet below and identify what the expected output is when the code is executed.

```
x = 8

y = 3


result = x == y * 2 + 2

print(result)
```

- ○ `True`
- ○ `False`
- ○ An error, because == is incorrectly used instead of =
- ○ Nothing, as the code contains an error

Read the code snippet below and identify what the expected output is when the code is executed.

```
a = 5

b = 10


if a == b:

    print("a is equal to b")

elif b == a:

    print("b is equal to a")

else:

    print("a is not equal to b")
```

- ○ `a is equal to b`
- ○ `b is equal to a`
- ○ `a is not equal to b`
- ○ Nothing, as the code contains an error

Read the code snippet below and identify what the expected output is when the code is executed.

```
a = True or 3/0

print(a)
```

- ○ `True`
- ○ Nothing, as the first operand in the or statement is already `True`
- ○ An error message related to division by zero
- ○ `False` (after displaying an error message)

Read the code snippet below and identify what the expected output is when the code is executed.

```
a = False

b = True

if a or b and not a:

    print("Condition met")

else:

    print("Condition not met")
```

- ○ No output, since there is an error with the logical operators
- ○ `Condition not met`
- ○ `Condition met`
- ○ No errors, but also no output

Skill Level Assessment - Error Message Comprehension

Based only on the error message below, which option best explains the cause of the error?

```
Traceback (most recent call last):

File "main.py", line 3, in <module>

    print(elements[5])

IndexError: list index out of range
```

- ○ The list `elements` has fewer than 6 elements

- ○ The list `elements` is not defined
- ○ The list is indexed starting at 1, so 5 is out of range
- ○ The list is empty and cannot be printed

Based only on the error message below, which option best explains the cause of the error?

```
Traceback (most recent call last):

File "main.py", line 1, in <module>

    def my_function()

    ^

SyntaxError: invalid syntax
```

- ○ The variable `my_function` hasn't been assigned a value
- ○ The function definition is missing a colon at the end
- ○ There's a problem with the indentation of the function
- ○ The function is missing a return statement

Based only on the error message below, which option best explains the cause of the error?

```
Traceback (most recent call last):

File "main.py", line 4, in <module>

    print(x)

NameError: name 'x' is not defined
```

- ○ The variable `x` has the wrong type assigned to it

- ◯ The code is attempting to access a variable from another function's scope
- ◯ The variable `x` hasn't been initialized or declared before being used
- ◯ The print statement must reference an index of `x`

Based only on the error message below, which option best explains the cause of the error?

```
Traceback (most recent call last):

File "main.py", line 2, in <module>

    int("one")

ValueError: invalid literal for int() with base 10: 'one'
```

- ◯ You cannot convert variables inside the `int()` function
- ◯ The string `"one"` is too long to be converted to an integer
- ◯ The `int()` function can only be used on floating point numbers
- ◯ The code is trying to convert a string that is not a valid number into an integer

Based only on the error message below, which option best explains the cause of the error?

```
Traceback (most recent call last):

File "main.py", line 5, in <module>
```

```
    open("file.txt", "r")

FileNotFoundError: [Errno 2] No such file or directory: 'file.txt'
```

- ◯ The file is open in another program and cannot be accessed
- ◯ The file `file.txt` does not exist in the current directory
- ◯ There is a syntax error when using the `open()` function to read `file.txt`
- ◯ The code is trying to write to a file that is set to read-only

Based only on the error message below, which option best explains the cause of the error?

```
Traceback (most recent call last):

File "main.py", line 3, in <module>

    my_list.remove(10)

ValueError: list.remove(x): x not in list
```

- ◯ The list `my_list` is empty, therefore nothing can be removed
- ◯ The value of `x` is not found in `my_list`, so it can't be removed
- ◯ You must assign the result of `.remove()` to a new variable
- ◯ The value of `10` is not found in `my_list`, so it can't be removed

Based only on the error message below, which option best explains the cause of the error?

```
Traceback (most recent call last):
```

```
File "main.py", line 3, in <module>

    data = {"name": "Alice", "age": 30}

    print(data["email"])

KeyError: 'email'
```

- ○ The dictionary `data` is not defined properly
- ○ The code is trying to access a key `'email'` that doesn't exist in the dictionary
- ○ The key `'email'` needs to be declared as a variable first
- ○ Dictionaries cannot store strings as keys

Skill Level Assessment - Natural Language Scenarios

You write a program to calculate the average of three numbers entered by the user. The code runs without crashing, but the result is slightly incorrect each time. What is the MOST likely cause for this, given the limited context?

- ○ A `SyntaxError` caused the wrong result to be stored.
- ○ A `TypeError` occurred when dividing numbers.
- ○ A logic error in how the average is computed.
- ○ An `IndexError` occurred due to incorrect list indexing.

You're debugging a program that initially throws a `NameError`. You fix it, but the same error pops up again after later code changes in subsequent code runs. What is the MOST likely explanation for this, given the limited context?

- ○ Your fix introduced a dependency that wasn't fully resolved elsewhere in the code
- ○ `NameError's` will reappear unless the variable type is explicitly declared
- ○ A similar issue is occurring in a different scope or function that wasn't originally part of the fix
- ○ `KeyError` and `NameError` are essentially interchangeable and often happen together

You're combining a number and a string using the + operator in Python, but it throws a `TypeError`. What concept do you need to understand to resolve this?

- ○ How Python distinguishes between different data types and converts them
- ○ The behavior of zero division and its impact on type operations
- ○ How Python prioritizes operators in mixed expressions
- ○ The correct way to define a function that adds strings and numbers

You write a function to calculate the average of a list of numbers. The function returns the correct result when you use it with small lists, but it crashes with an

`IndexError` when used on a very large list. What is the MOST likely reason for this, given the limited context?

- ○ The error arises from recursion depth, not from list indexing directly
- ○ The function contains assumptions about list size or structure that don't hold for all inputs
- ○ Python's internal list functions don't support large datasets by default
- ○ An `IndexError` indicates a problem with your computer's memory allocation

You're writing a program that uses a dictionary to store user preferences. You try to access a key that you *know* exists in the dictionary, but your code throws a `KeyError`. What is the MOST likely reason for this, given the limited context?

- ○ The key is actually a string, but you're trying to access it with an integer
- ○ There's a typo in the key you are trying to access (e.g., `"User"` vs `"user"`), or the case sensitivity is different than expected
- ○ Dictionaries can sometimes fail to locate keys due to internal hashing bugs or collisions
- ○ The key might have been altered or removed elsewhere in the code

You're iterating through a list of strings and calling a method on each, but receive an `AttributeError`. What is the MOST likely reason for this, given the limited context?

- ○ The strings in your list are actually integers disguised as strings
- ○ You're attempting to access or call something (a method or attribute) that isn't available for the type of object you're working with
- ○ `AttributeError` only occurs when using recursion
- ○ The `for` loop is incorrectly structured and is causing unexpected behavior

You're writing a program that uses recursion to calculate the factorial of a number. The program works correctly for small numbers, but crashes with a `RecursionError` on large inputs. What's the MOST likely reason for this, given the limited context?

- ○ `RecursionError` is usually caused by defining `for` loops incorrectly
- ○ Python's recursion depth limit has been exceeded, meaning the function called itself too many times
- ○ The base case is incorrectly written, causing infinite recursion
- ○ Factorial can't be computed for large numbers in Python due to memory limits

Skill Level Assessment - Varying Difficulty & Scope

Review the function below and determine the logical error causing an incorrect result, if any.

```
def find_max(numbers):

    max_val = 0

    for num in numbers:

        if num > max_val:

            max_val = num

    return max_val


result = find_max([-10, -5, -3])
```

- ○ It raises an exception due to negative values being present in the list.
- ○ It incorrectly returns the wrong number instead of the maximum value due to improper initialization of `max_val`
- ○ There is no logical error present; the code correctly assigns the value of `-3` to the `result` variable.
- ○ The `for` loop is structured incorrectly as it does not iterate through all of the array's values.

Examine the following `Rectangle` class definition. What is the issue in this code, if any?

```
class Rectangle:

    def __init__(self, width, height):

        self.width = width

        self.height = height


    def area(self):

        return width * height
```

```
rect = Rectangle(3, 4)

print(rect.area())
```

- ○ The `area` method is missing the `self` parameter in its definition
- ○ There is no error; the code is both syntactically and semantically correct.
- ○ The call to `Rectangle(3, 4)` is invalid because the constructor expects 3 arguments, including `self`
- ○ The attributes `width` and `height` are not prefixed with `self.` inside the `area` method.

Read the code snippet below. The `process_data` function can sometimes encounter an error during execution. Which option correctly identifies the cause and suggests a valid fix?

```
def process_data(data_list):

    total = 0

    for item in data_list:

        try:

            total += int(item)

        except ValueError:

            print(f"Skipping non-integer value: {item}")


    average = total / len(data_list)

    return average


# Example usage

data = ["10", "20", "thirty", "", "40"]
```

```
result = process_data(data)

print("The average is:", result)
```

- ◯ Remove `try-except` block; it's unnecessary with proper input validation
- ◯ Convert empty strings to zero before adding to `total`
- ◯ Ensure division by a non-zero number by checking if `data_list` is not empty before calculating the average
- ◯ All of the above are valid fixes

Analyze the following Python code. What error will be raised when it is executed?

```
def get_value(d, key):

    return d[key]


d = {"a": 1, "b": 2}

print(get_value(d, "c"))
```

- ◯ `TypeError: 'dict' object is not callable`
- ◯ `KeyError: 'c'`
- ◯ `NameError: name 'd' is not defined`
- ◯ `SyntaxError: invalid syntax`

In the code snippet below, what is the mistake that prevents the correct evaluation of the condition?

```
x = "10"

if x = 10:
    print("x is ten")
```

- The print statement is incorrect
- Assignment operator is used insted of comparison operator
- The condition should be wrapped in parentheses
- The variable `x` is compared to an integer instead of a string

The code snippet below raises an error and does not compute the sum of the list correctly. Choose the option that both fixes the error and ensures the correct sum (10) is printed on line 6.

```
1: def sum_list(lst):

2:     total = 0

3:     for i in lst:

4:         total += i

5:     return total

6: print(sum_list([1, 2, "3", 4]))
```

- Change the function name to `total_sum`
- Cast each element to an `int` before addition
- Remove the `for` loop
- Remove the string `"3"` from the list

Examine the following code. What type of error will occur when the code is executed?

```
class MyClass:

    def __init__(self, value):

        self.value = value

    def display():

        print(self.value)


obj = MyClass(10)

obj.display()
```

- ◯ NameError: name 'self' is not defined
- ◯ TypeError: MyClass.display() takes 0 positional arguments but 1 was given
- ◯ AttributeError: 'MyClass' object has no attribute 'display'
- ◯ SyntaxError: invalid syntax

Post Examination

Take a moment to reflect on your responses. Approximately how many of the 16 multiple choice questions do you think you got right? Please provide your best guess as a number from 0 to 16.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Select Number of Correct Questions																	

Powered by Qualtrics