ISSN NO: 2770-0003
Date of Publication: 08-03-2022

Learn how to prevent errors while writing Python-code

Amirov Islombek Dilshodbek son,
Ahmadaliyev Doniyorbek Kambaraliyevich
Andijan State University, Department of Information Technology Presidential School in Andijan
E-mail: islombek.a.d.1606@gmail.com

Annotation: Python is widely used by many programmers around the world to create a variety of software applications. The most interesting thing about Python is that it is one of the most programmed languages compared to other programming languages with its simple code and extensive capabilities. That is, you do not have to declare the types of variables when creating a variable in python. It is selected automatically. In addition, error handling allows you to display and manage multiple errors of the same type. Incomplete parentheses and quotation marks in Python can cause various types of errors. Most programmers make mistakes during the coding process, and programmers detect some errors when Python executes a program. In the process of writing code, in some cases, Python detects an error and the program stops. In this case, we will look at the errors, and since there are many types of errors in Python, we will cover most of the errors in this article.

Keywords: Types of errors in Python, Python Logical errors, Python Exception, Prevention of errors in Python. Nomenclature:
SyntaxError:
Eol va Eof :
IndentatinError:
NameError:
ValueError:
ZeroDivisionError:
TypeError:
IndexError:
Run timeError:

Introduction:

When we write code in the Python programming language, there are types of errors depending on the code we write. These are mainly Syntax errors, Placement errors, Program execution errors, Logical errors, Exceptions errors. Each error has its own name.

SyntaxError is one of the most common errors in our software and is usually caused by not following the rules when writing code. Programming environments detect such errors before the program is executed and signal to the programmer. Python does not run a program that has a syntax error.

Syntax also provides additional information in the error: The EOL error is one of the syntax errors, which is mainly caused by not closing the quotation mark or a quotation mark at the end of the line. The EOF error is caused by not closing the parentheses at the end of the function. IndentationEror- This is one of the errors caused by not leaving space in special places of our code. If we want to omit a place, we need to press the TAB (keyboard) or (PROBEL) button in these cases, and in our current code we will show you in more detail about these errors.

ISSN NO: 2770-0003
Date of Publication: 08-03-2022

Figure 1

```
print" SyntaxsError xatoligi"
```

```
  File "<ipython-input-5-241c43af7f91>", line 1
    print" SyntaxsError xatoligi"
          ^
SyntaxError: invalid syntax
```

```
print("EOL xatoligi)
```

```
  File "<ipython-input-6-6091d607a711>", line 1
    print("EOL xatoligi)
                        ^
SyntaxError: EOL while scanning string literal
```

```
print("EOF xatoligi"
```

```
  File "<ipython-input-7-4df0f965802f>", line 1
    print("EOF xatoligi"
                        ^
SyntaxError: unexpected EOF while parsing
```

```
dasturlash_tillari=["Python","PHP","JS","C+","IOS"] # Dasturlash degan o'zgaruvchiga Nomlarni yozib chiqamiz.
for dasturlash in dasturlash_tillari:
print(f" Bu {dasturlash} dunyodagi eng ommalashgan dasturlash tillari hisoblanadi.")
```

```
  File "<ipython-input-1-109db80c110a>", line 3
    print(f" Bu {dasturlash} dunyodagi eng ommalashgan dasturlash tillari hisoblanadi.")
    ^
IndentationError: expected an indented block
```


This image shows errors:

Unlike syntax errors, in the process of writing code in Python, we introduce the types of errors that run the program and stop the process (Run Time Error).

NameError- This is usually an error in typing the name of a function or variable while writing code on a computer. ValueError is one of the errors that can lead to incorrect values for a function, such as adding text to numbers, assigning values to methods, and typing some code. When performing ZeroDivisionError- mathematical operations, an error occurs as a result of dividing numbers ("to the number 0"). TypeError - An error occurred as a result of incorrect information about the type or type of code we wrote. As a programmer, when we write code, we get an IndexError error when we refer to lists incorrectly.

for example:

ISSN NO: 2770-0003
Date of Publication: 08-03-2022

Figure 2

```
x,y=45,0
print(x/y)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-7-0ccf6f3042cb> in <module>
      1 x,y=45,0
----> 2 print(x/y)

ZeroDivisionError: division by zero
```

```
yil=input("Yilingizdi kiriting") and print(f"Siz {2021-yil} yoshdasiz")
```

```
Yilingizdi kiriting2000

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-10-e70cc4448db2> in <module>
----> 1 yil=input("Yilingizdi kiriting") and print(f"Siz {2021-yil} yoshdasiz")

TypeError: unsupported operand type(s) for -: 'int' and 'str'
```

```
otm_lar=["ADU","Namdu","FarDu","SamDu"] # Dasturlashda tartib raqam 0 dan boshlanadi
# shu sababdan ro'yxatda yo'q elementdi chaqirilganligi sababli ushbu xato ko'rsatadi.
print(otm_lar[4])
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-6-9792b7fe111e> in <module>
      1 otm_lar=["ADU","Namdu","FarDu","SamDu"]
----> 2 print(otm_lar[4])

IndexError: list index out of range
```

One of the mistakes that can be made when writing a program algorithm when performing mathematical operations is called Logical Errors. Logical errors do not prevent us from stopping when writing a program, but they do not give the correct result at the end of the program. Logical errors are one of the most common and difficult errors. Python detects logical errors and does not show you errors, and does not produce unexpected results. In the future, when we create big programs, it will have a lot of functions, but if we do not find logical errors in time, we will need to check a lot of our written code. The following figure does not illustrate that we have made brief logical errors in our current code.

Figure 3

```
son=float(input("Sonni kiriting?")) # Bu yozgan kodimiz o'z o'zidan xato chunki
# matematik mantiqiy xatoga yo'l qo'ydik yani ildizni qavsga olmaganimiz sabab.
ildiz=son**1/2
print(f"{son} ning ildizi {ildiz} ga teng.")
```

```
Sonni kiriting?9
9.0 ning ildizi 4.5 ga teng.
```

Leaving or not leaving spaces in our code can also be logical errors. For this reason, most bug fixes are detected and updated frequently when the software is exported to the world market. As an example, our phone asks for updates from time to time. This is because bugs are found and updated by programmers.

ISSN NO: 2770-0003
Date of Publication: 08-03-2022

Figure 4

```
raqamlar=list(range(1,5))
for raqam in raqamlar:
    print(raqam)
    print("1 dan 4 gacha raqamlar ro'yxati.")
```

```
1
1 dan 4 gacha raqamlar ro'yxati.
2
1 dan 4 gacha raqamlar ro'yxati.
3
1 dan 4 gacha raqamlar ro'yxati.
4
1 dan 4 gacha raqamlar ro'yxati.
```

In this code, we included the command to end the list of numbers from one to four, but the result we expected did not work, because if we type the last print ("list of numbers from 1 to 4") without leaving a space, we get the expected result.

Exceptions - This is used to prevent our program from crashing in the event of a Python error.

Figure 5

```
yosh=input("Yilingizdi kiriting?")
try:
    yosh=int(yosh)
    print(f"siz {2021-yosh} yil ekansiz. ")
except:
    print("Siz yoshingizdi butun son shaklida kiritmadingiz?")
print("Dasturimiz nixoyasiga yetdi.")
```

```
Yilingizdi kiriting?21.0
Siz yoshingizdi butun son shaklida kiritmadingiz?
Dasturimiz nixoyasiga yetdi.
```

One of the conveniences of the try-except operator is that instead of typing errors in Python, we can write texts that we can understand independently. Prevents our program from crashing due to errors in Python, in this code.

Figure 6.

```
while True:
    yosh= input("Yoshingiz kiriting:")
    if yosh.isdigit():
        yosh=int(yosh)
        break
print(f"Siz {2021-yosh} yilda tug'ilgansiz")
```

```
Yoshingiz kiriting:12.0
Yoshingiz kiriting:17.0
Yoshingiz kiriting:19.0
Yoshingiz kiriting:21
Siz 2000 yilda tug'ilgansiz
```

It is advisable to use if-else-while loops to avoid errors in
Python. Results The results of our article this summer show that each mistake has its drawbacks. Let's take a brief look at the codes we've written so far and explain the results. Figure 2 (2) (ValueError) In this code, we converted the number entered by the user in the first line to the integer int (). To avoid errors, we need to change the int () function to the float () function or enter the desired number as an integer. Figure 3 Logical

ISSN NO: 2770-0003
Date of Publication: 08-03-2022

errors We were asked to enter a number but the result was four halves of the root of nine. When we analyze our code, the second line makes an error in calculating the root because it does not include 1/2 parentheses, because the computer raises the entered number to the first level and then divides it by two, which corrects the error and eliminates the error. Figure 5 In this case, the computer will ask you to enter your year, and the computer will use the int () function (try). If you enter the correct value, it will calculate the year you entered. If an exception occurs ("Did you not enter your year as an integer?"), The program will end without stopping ("The program is over"). The convenience of this (try-except) operator is that instead of the incomprehensible errors that python displays the number we enter, we can display more plaintext. Instead of reviewing these errors one by one, we'll give you a brief overview to help you understand the consequences and the codes we've written.

Conclusion:

As a programmer we have to conclude from this that the code writing process is simple and errors can occur when we write any, especially large programs. This can cause it to not work or the program not to do what it needs to do. There are many causes of errors, a programmer can make a mistake in writing code by finding a solution to a problem and compiling its algorithm. Each code should be carefully checked to ensure that no errors occur during program writing. For example, write the name of a variable in a clearer way, or be sure to put a colon in the title of a complex sentence. If you are new to the world of programming, whether you are a junior or a senior programmer, try to write as accurately as possible. Whether you work as a programmer or a government employee in your future endeavors, make it a habit not to make mistakes in all areas.

References:

1. https://www.tutorialsteacher.com
2. http://net-informations.com/python/err/eol.htm
3. https://www.stechies.com/convert-list-string-python/
4. Basics of Python programming.
5. Python Basics (English) .pdf
6. Python Fastlane Crash Course
7. Python Pocket Reference (en)