

VAULT OF CODES

ASSIGNMENT-1

Task1:

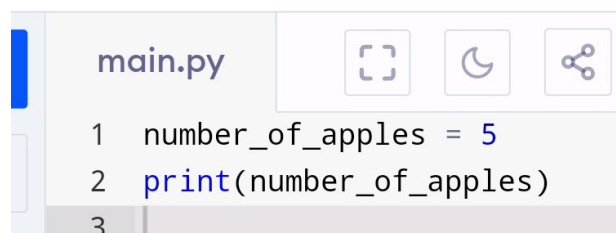
Review the following codes, find and fix errors and explain the error

1.

Error Code:

```
number_of_apples = 5  
print(number_of_apple)
```

CORRECTEDCODE:

A screenshot of a code editor window titled 'main.py'. It contains three lines of Python code: line 1 is 'number_of_apples = 5', line 2 is 'print(number_of_apples)', and line 3 is empty. The editor has icons for file operations (new, open, save, share) in the top right corner.

```
main.py  
1 number_of_apples = 5  
2 print(number_of_apples)  
3
```

Explanation:

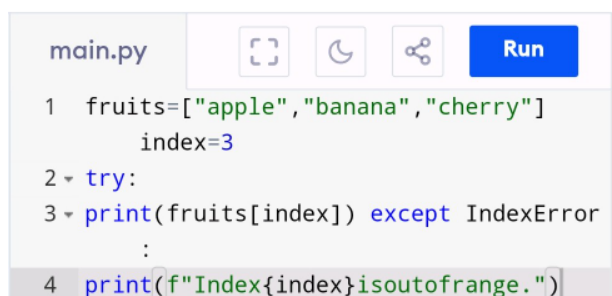
In the above error code ,especially we are trying to print the variable "number_of_apple", which does not exist because you have defined the variable as "number_of_apples" . So, name error will occur.Name error indicates access to an undeclared variable.

2.

Error Code:

```
fruits=["apple","banana","cherry"]  
print(fruits[3])
```

Corrected code:

A screenshot of a code editor window titled 'main.py'. It contains four lines of Python code: line 1 is 'fruits=["apple","banana","cherry"]', line 2 is 'index=3', line 3 is 'try:', and line 4 is 'print(fruits[index]) except IndexError:'. The editor has icons for file operations (new, open, save, share) and a 'Run' button in the top right corner.

```
main.py  
1 fruits=["apple","banana","cherry"]  
  index=3  
2 try:  
3 print(fruits[index]) except IndexError:  
4 print(f"Index{index}isoutofrange.")
```

Explanation:

IndexError is raised when attempting to access an index which is outside the valid range of a sequence.

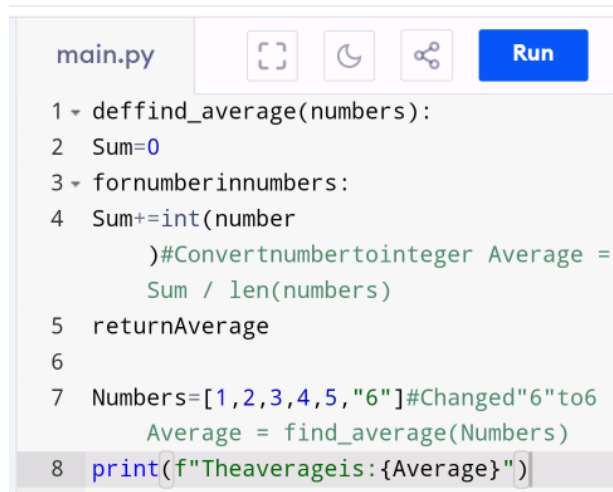
In your example, the list "fruits" has only 3 elements:
--"apple" is at index 0,--"banana" is at index 1, -- "cherry" is at index 2
Index 3 doesn't have any element, So it causes index error.

3.Error code:

```
def find_average(numbers):  
    Sum=0  
    for number in numbers:  
        Sum  
        += number  
    Average=Sum/len(numbers) return  
    Average
```

```
Numbers=[1,2,3,4,5,"6"]  
Average=find_average(Numbers)  
print(f"The average is:{Average}")
```

Corrected code:



```
main.py  [Icons]  Run  
1 def find_average(numbers):  
2     Sum=0  
3     for number in numbers:  
4         Sum+=int(number  
           )#Convert number to integer Average =  
           Sum / len(numbers)  
5     return Average  
6  
7     Numbers=[1,2,3,4,5,"6"]#Changed "6" to 6  
           Average = find_average(Numbers)  
8     print(f"The average is:{Average}")
```

Explanation:

The error in the code is that the list "numbers" contains a string ("6") instead of an integer (6). When the function tries to add the string "6" to the sum, it throws a TypeError because you can't add a string to an integer.

- The `int(number)` conversion ensures that each element in the list is treated as an integer, even if it's originally a string.

With these changes, the function should work correctly and print the average of the numbers in the list.

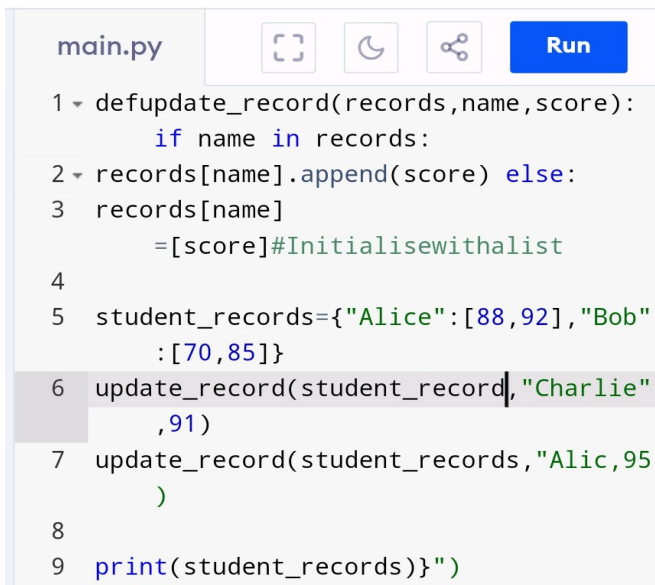
4.

Error code:

```
def update_record(records, name, score):  
    if name in records:  
        records[name].append(score)  
    else:  
        records[name] = score
```

```
student_records = {"Alice": [88, 92], "Bob": [70, 85]}  
update_record(student_records, "Charlie", 91)  
update_record(student_records, "Alice", 95)  
print(student_records)
```

Corrected code:



```
main.py  [Icons] [Run]  
1 def update_record(records, name, score):  
    if name in records:  
2 records[name].append(score) else:  
3 records[name]  
    = [score] # Initialisewithalist  
4  
5 student_records = {"Alice": [88, 92], "Bob"  
    : [70, 85]}  
6 update_record(student_record, "Charlie"  
    , 91)  
7 update_record(student_records, "Alic, 95"  
    )  
8  
9 print(student_records)}")
```

Output:

```
{'Alice': [88, 92, 95], 'Bob': [70, 85], 'Charlie': [91]}
```

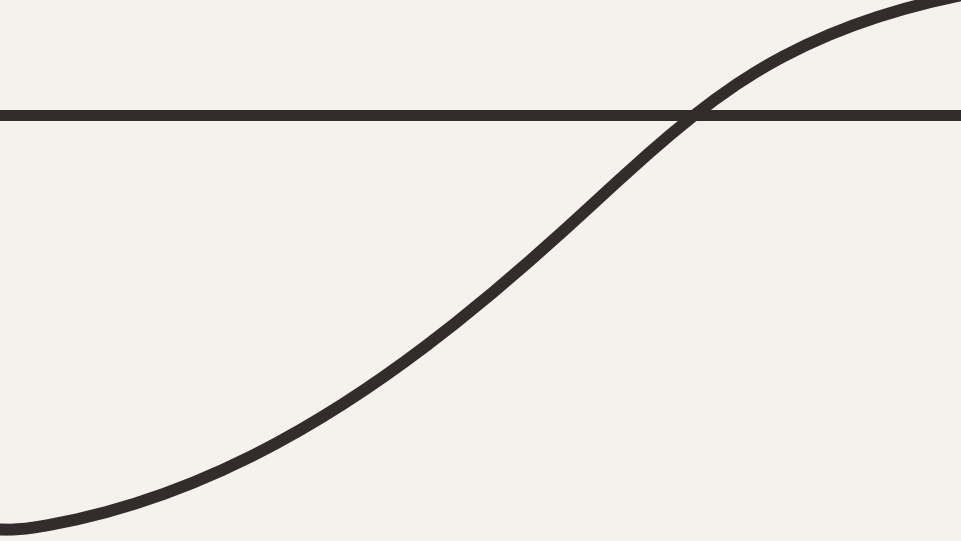
Explanation :

The error is in the else block of the `update_record` function. When a new name is encountered, the score is assigned as the value without being wrapped in a list. This causes an error when trying to append a score to a non-list value.

In the else block, I changed `records[name] = score` to `records[name] = [score]` to ensure that the value associated with the name is always a list, even if

it's the first score being added.

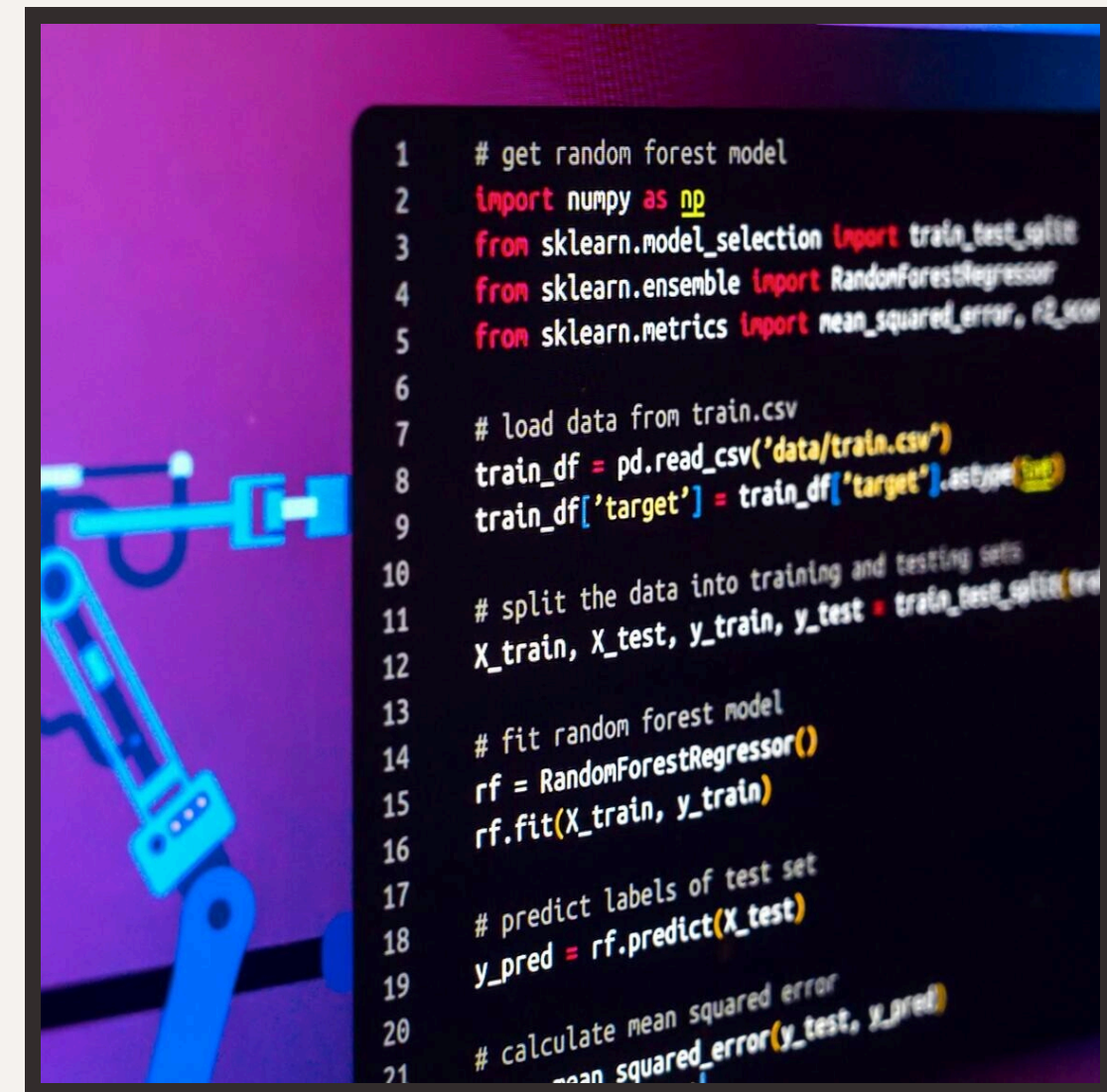
Now,whenanewnameisencountered,thescoreiscorrectlywrappedin a list, allowing subsequent scores to be appended without errors.



Introduction and history of python and it's functions and modules

History of Python

Python was created by **Guido van Rossum** and first released in **1991**. It was designed to be easy to read and write, focusing on **code readability**. Over the years, Python has evolved significantly, becoming one of the most popular programming languages worldwide.



```
1 # get random forest model
2 import numpy as np
3 from sklearn.model_selection import train_test_split
4 from sklearn.ensemble import RandomForestRegressor
5 from sklearn.metrics import mean_squared_error, r2_score
6
7 # load data from train.csv
8 train_df = pd.read_csv('data/train.csv')
9 train_df['target'] = train_df['target'].astype(int)
10
11 # split the data into training and testing sets
12 X_train, X_test, y_train, y_test = train_test_split(train_df[['x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7', 'x8', 'x9', 'x10', 'x11', 'x12', 'x13', 'x14', 'x15', 'x16', 'x17', 'x18', 'x19', 'x20', 'x21', 'x22', 'x23', 'x24', 'x25', 'x26', 'x27', 'x28', 'x29', 'x30', 'x31', 'x32', 'x33', 'x34', 'x35', 'x36', 'x37', 'x38', 'x39', 'x40', 'x41', 'x42', 'x43', 'x44', 'x45', 'x46', 'x47', 'x48', 'x49', 'x50', 'x51', 'x52', 'x53', 'x54', 'x55', 'x56', 'x57', 'x58', 'x59', 'x60', 'x61', 'x62', 'x63', 'x64', 'x65', 'x66', 'x67', 'x68', 'x69', 'x70', 'x71', 'x72', 'x73', 'x74', 'x75', 'x76', 'x77', 'x78', 'x79', 'x80', 'x81', 'x82', 'x83', 'x84', 'x85', 'x86', 'x87', 'x88', 'x89', 'x90', 'x91', 'x92', 'x93', 'x94', 'x95', 'x96', 'x97', 'x98', 'x99', 'x100'], train_df['target'], test_size=0.2, random_state=42)
13
14 # fit random forest model
15 rf = RandomForestRegressor()
16 rf.fit(X_train, y_train)
17
18 # predict labels of test set
19 y_pred = rf.predict(X_test)
20
21 # calculate mean squared error
22 mean_squared_error(y_test, y_pred)
```



```

the text runs across the top
persisted properties
<html> <error>
<html> <p style="font-weight:bold;">HTML font
<html> <body style="background-color:yellowgreen"
<html> <text - :200px;"> <.todolistid = data.todolistid
/ Non - text - :200px;">persisted properties
<html> <errorMessage = ko , observable()
style="color:orange;">HTML font code is
function todoitem(data) ; <html> <error>
var self = this <html> <error>
data = data || {} <html> <error>
/ Non - persisted properties
<html> <errorMessage = text - :200px;">
<p style="font-weight:bold;">HTML font code is
<body style="background-color:yellowgreen"
text - :200px;"> <.todolistid = data.todolistid
- text - :200px;">persisted properties
<errorMessage = ko , observable()

```


One of Python's standout attributes is its **simple syntax**, which resembles natural language. This makes it an ideal choice for beginners. The use of **indentation** for block delimiters enhances readability and encourages best coding practices.

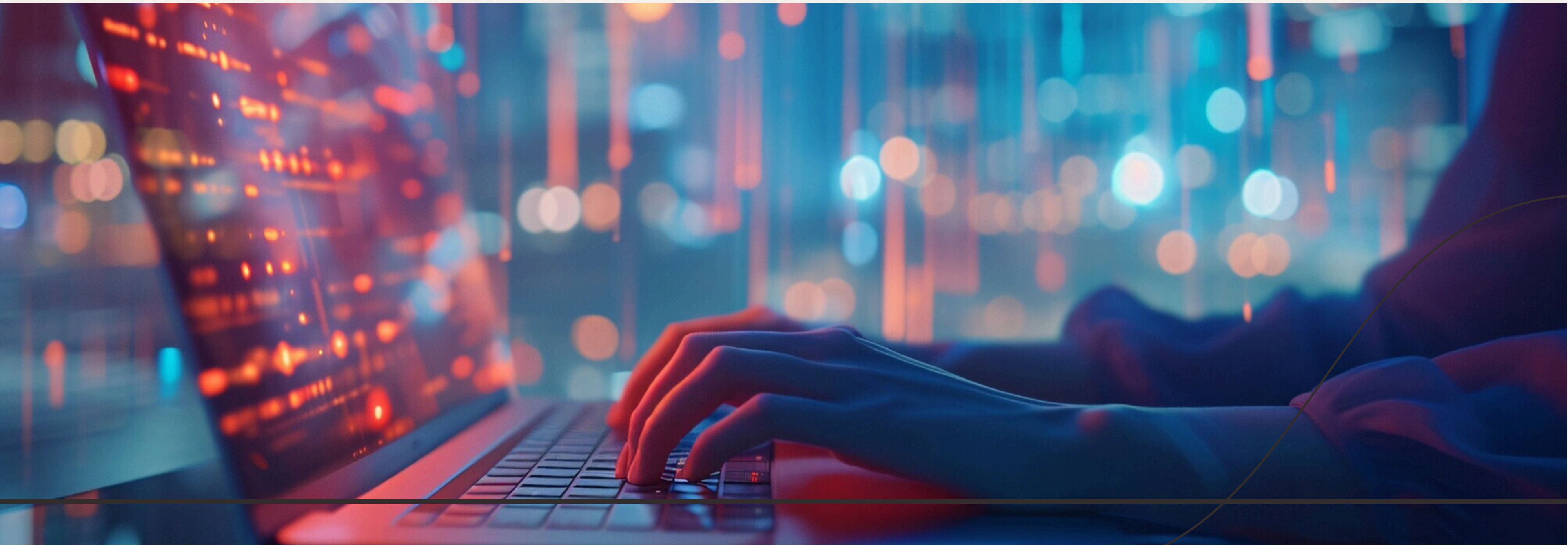




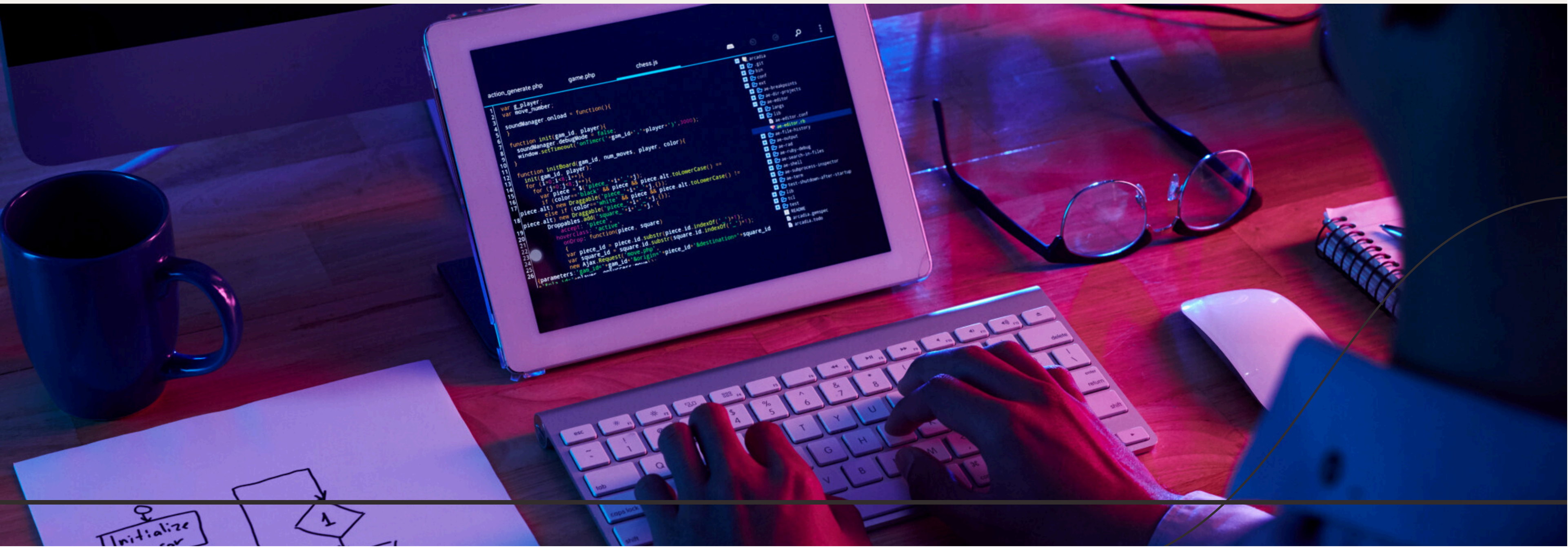
Functions in Python

Functions are fundamental in Python, allowing for **code reusability** and organization. With built-in functions and the ability to create custom ones, Python promotes modular programming. Understanding functions is essential for effective coding in Python.

Python includes a variety of **modules** that extend its functionality, such as **NumPy** for numerical computations, **Pandas** for data manipulation, and **Matplotlib** for data visualization. These modules empower developers to tackle diverse tasks efficiently.



Python has become a dominant language in **data science** due to its powerful libraries and tools. Its capabilities in **machine learning**, **data analysis**, and **visualization** make it indispensable for data professionals seeking to extract insights from data.



Future of Python



As technology evolves, so does Python. Its versatility and adaptability ensure its relevance in emerging fields like **artificial intelligence** and **cloud computing**. The future looks bright for Python, with continuous enhancements and a growing user base.