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:

main.py

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

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:

main.py Run

```
fruits=["apple","banana","cherry"]
index=3
try:
print(fruits[index]) except IndexError
:
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:

```
deffind_average(numbers):
   Sum=0
   fornumberinnumbers: Sum
      += number
   Average=Sum/len(numbers) return
   Average

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

Corrected code:

main.py Run

```
deffind_average(numbers):
Sum=0
fornumberinnumbers:
Sum+=int(number
    )#Convertnumbertointeger Average =
    Sum / len(numbers)
returnAverage

Numbers=[1,2,3,4,5,"6"]#Changed"6"to6
    Average = find_average(Numbers)
print(f"Theaverageis:{Average}")
```

Explanation:

The error in the code is that the list "numbers" contains a string ("6") insteadofaninteger(6).Whenthefunctiontriestoaddthestring"6"to the sum, it throws a TypeError because you can't add a string to an integer.
- Theint(number)conversionensuresthateachelementinthelistis treated as an integer, even if it's originally a string.

Withthesechanges,thefunctionshouldworkcorrectlyandprintthe average of the numbers in the list.

4.

Error code:

```
defupdate_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 Run

```
1 defupdate_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 :

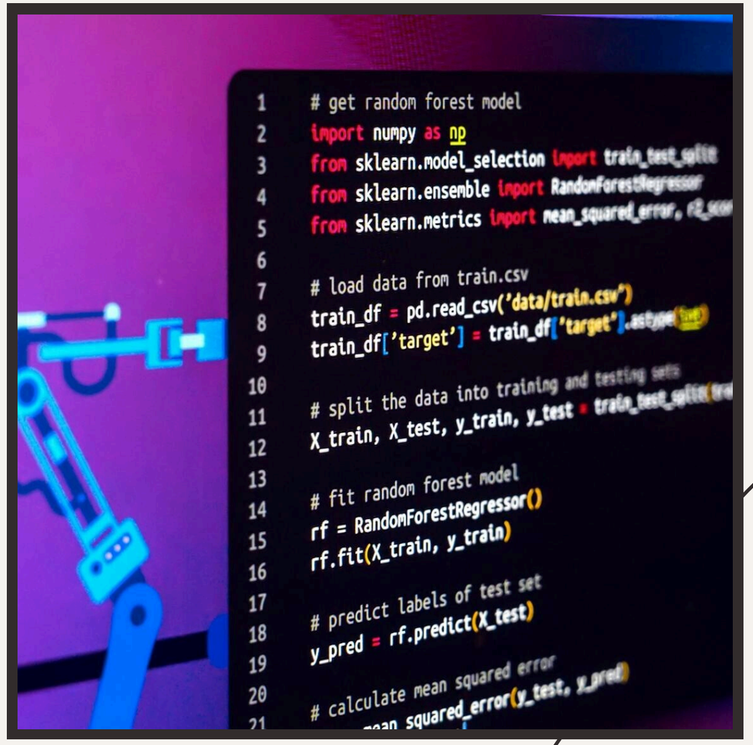
heerrorisintheelseblockoftheupdate_recordfunction.Whenanew name is encountered, the score is assigned as the value without being wrappedinalist.Thiscausesanerrorwhentryingtoappendascoretoa non-list value.
In the else block, I changed records[name] = score to records[name] = [score]toensurethatthevalueassociatedwiththenameisalwaysalist, 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

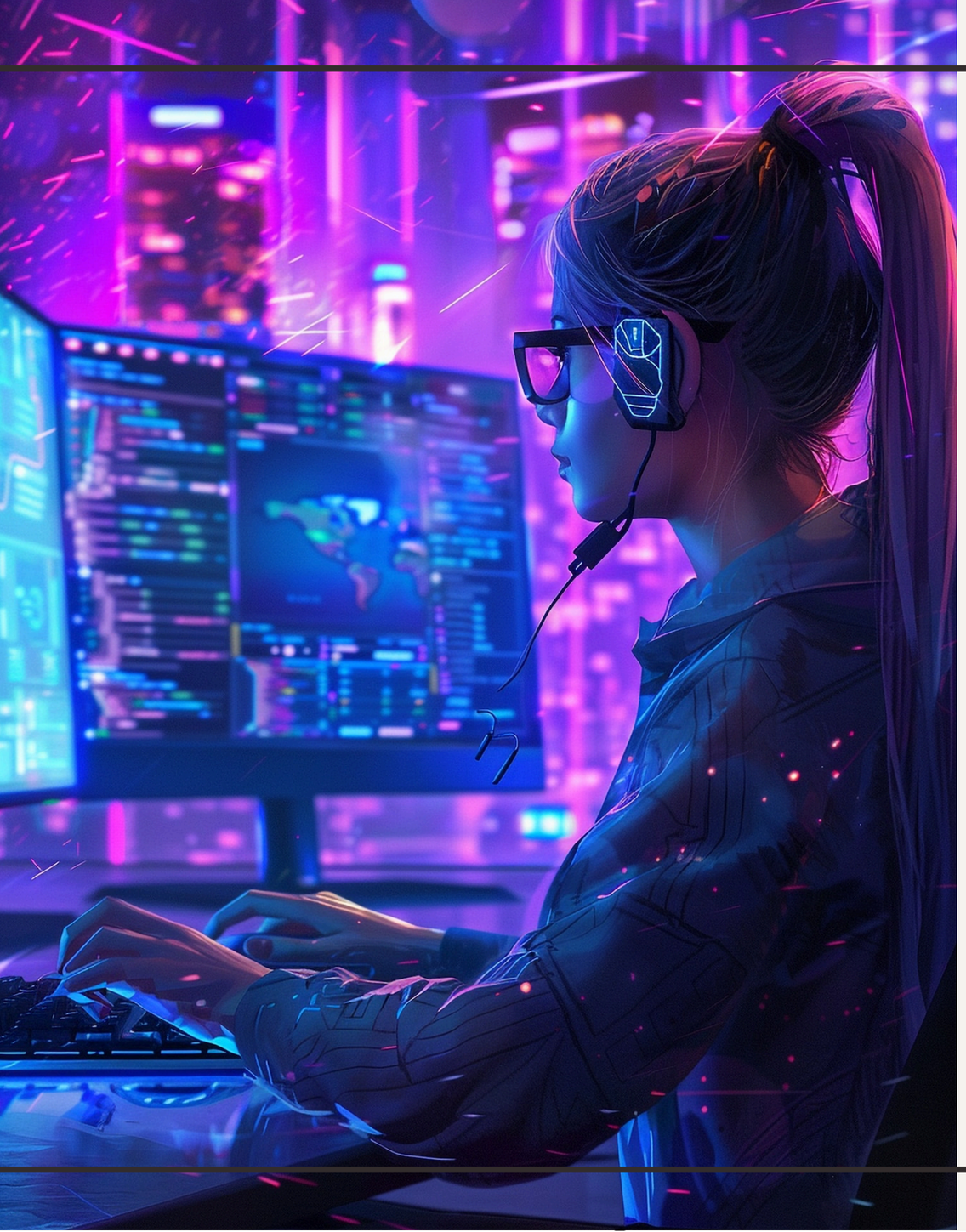
Python was created by **Guido van Rossum** and first released in **1991**. It was designed to be easy to read and write, focusing on **code readability**. Over the years, Python has evolved significantly, becoming one of the most popular programming languages worldwide.



Features of Python

Python boasts several **key features** that contribute to its popularity, such as **simplicity**, **interactivity**, and a vast **standard library**. These features enable developers to write clear and concise code, fostering productivity and collaboration.

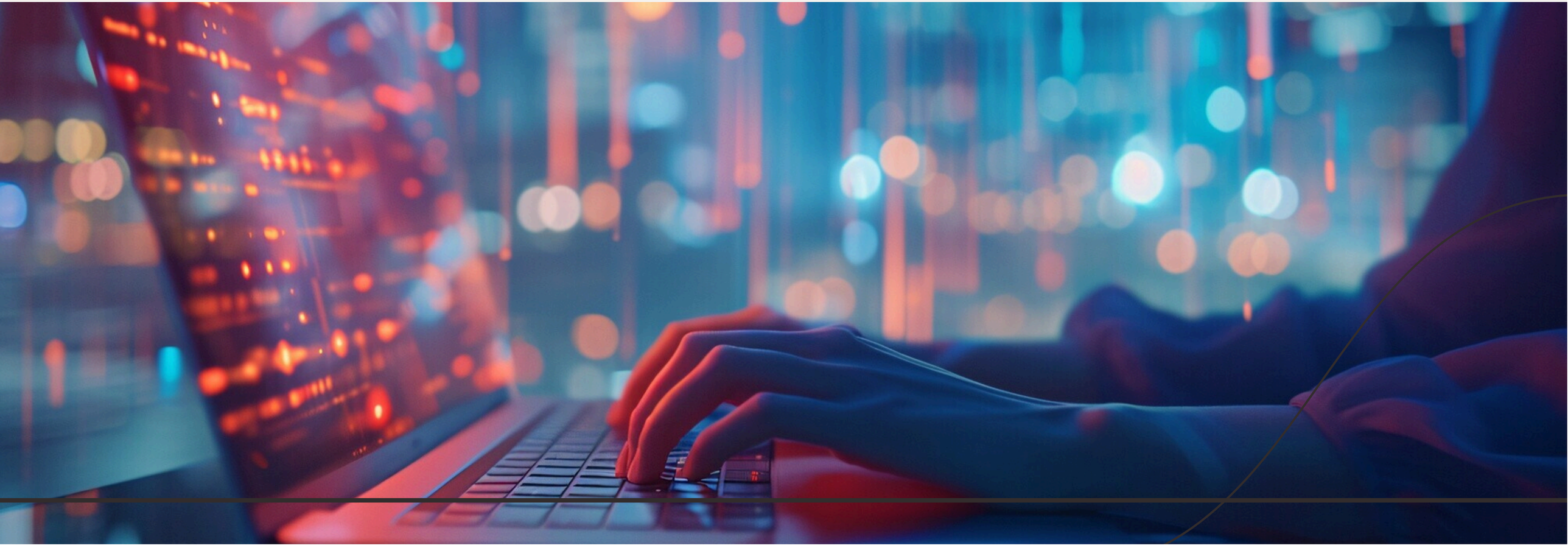
One of Python's standout attributes is its **simple syntax**, which resembles natural language. This makes it an ideal choice for beginners. The use of **indentation** for block delimiters enhances readability and encourages best coding practices.



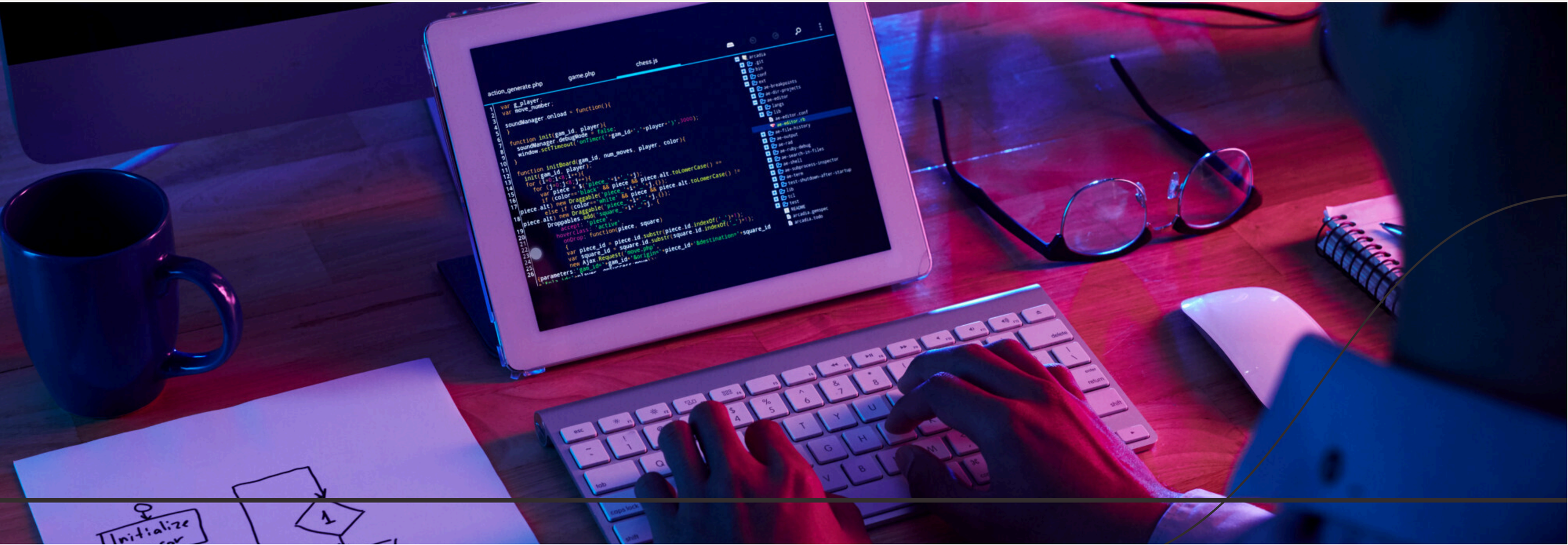
Functions in Python

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.