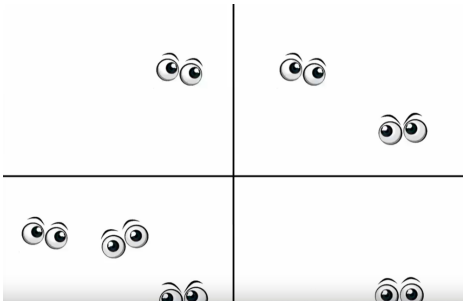
Building Smarter Python Programs: Decisions, Loops, and Error Handling

1 Introduction: The Story of the Eyes

Imagine you're looking at a picture. It's a grid, and inside each of the four cells, there's an eye staring back at you. What do you see? Are they all the same eyes, or are they different? Are they human eyes or animal eyes? Are they looking in the same direction?



Your mind probably started weaving a little story. Maybe they're suspicious eyes, or maybe they're surprised. Maybe they're all watching you. Your mind is naturally deciding, sorting, and connecting based on what you see. It is automatically looping through the various scenarios you can imagine.

That's precisely what we're going to learn how to do in Python: make decisions, repeat actions, and handle unexpected situations, just like your mind does when you look at a picture.

2 Python Thinks Like You: The Power of Decisions

Our minds make decisions constantly. Should we wear a jacket? Should we cross the street? Programming languages, like Python, need to make decisions too. That's why they include logic constructs that allow programs to "choose" different paths based on certain conditions, just like you choose a story for the eyes.

Note: The symbol ␣ in code represents a **space character**. It is used to explicitly indicate spaces, you might find them in your python code in this document.

2.1 The if Statement: Making Choices

Let's start by taking an age from the user. We need to define it like: `age = 40`. Instead of us directly defining the value of the variable, let's make the user define it using `input()`.

Now, let's say we want to validate the age someone has provided. We don't want negative ages, and we don't want zero. If we need to check the value of `age`, that's where the `if` statement comes in.

```
age = int(input("Enter␣the␣age:␣"))  # Get age from user, convert
    to integer

if age < 0:
    print("Your␣age␣cannot␣be␣negative")
```

2.2 The elif Statement: Adding More Conditions

But what if the age is zero? This is where `elif` (else if) comes in.

```
age = int(input("Enter␣the␣age:␣"))  # Get age from user, convert
    to integer

if age < 0:
    print("Your␣age␣cannot␣be␣negative")
elif age == 0:
    print("Age␣cannot␣be␣zero")
```

2.3 The underlineelse Statement: The Default Path

Finally, if the age is not negative and not zero, it must be a valid age. We can use `else` to handle this default case.

```
age = int(input("Enter␣the␣age:␣"))  # Get age from user, convert
    to integer

if age < 0:
    print("Your␣age␣cannot␣be␣negative")
elif age == 0:
    print("Age␣cannot␣be␣zero")
else:
    print("Age␣is␣valid")
```

2.4 Combining Decisions: Let us calculate your birth year!

```python
age = int(input("Enter the age: "))  # Get age from user, convert to integer

if age < 0:
    print("Your age cannot be negative")
elif age == 0:
    print("Age cannot be zero")
else:
    print("Age is valid")
    current_year = int(input("Enter the current year: "))
    birth_year = current_year - age
    print(f"You were born in {birth_year}")
```

The line $print(f"You were born in birth_year")$ is an example of an

2.5 Example to find : Pass or Fail

Listing 1: Pass or Fail Decision

```python
# Input: Student's score
score = int(input("Enter the student's score: "))

# Checking the grade
if score >= 50:
    print("Pass")
elif score >= 40:
    print("Supplementary Exam")
else:
    print("Fail")
```

Explanation

- If the score is **50 or above**, the student **passes**.
- If the score is **between 40 and 49**, they need to take a **supplementary exam**.
- If the score is **below 40**, they **fail**.

2.6 Python Code for determining grading system

Listing 2: Grade Assignment

```python
# Input: Student's score
score = int(input("Enter the student's score: "))

# Checking the grade
if score >= 90:
    print("Grade: A")
elif score >= 80:
    print("Grade: B")
elif score >= 70:
    print("Grade: C")
```

```
elif score >= 60:
    print("Grade: D")
else:
    print("Grade: F")
```

2.7 Explanation

- If the score is **90 or above**, the grade is **A**.
- If the score is between **80 and 89**, the grade is **B**.
- If the score is between **70 and 79**, the grade is **C**.
- If the score is between **60 and 69**, the grade is **D**.
- If the score is **below 60**, the grade is **F**.

3 Looping: Doing Things Again and Again

Remember how your mind looped through the four eyes, considering the possibilities? In programming, we use loops to repeat actions. Let's take an example. Say we have a list of numbers, we want to check if they are negative or positive. A loop would be useful.

3.1 The for Loop: Iterating through Items

Let's say we have a list of numbers:

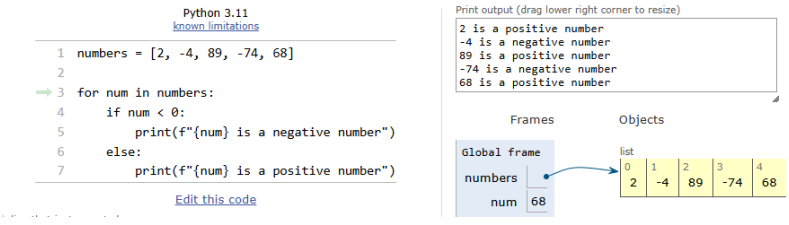
```
numbers = [2, -4, 89, -74, 68]
```

We can use a `for` loop to go through each number and decide whether it is positive or negative.

```
numbers = [2, -4, 89, -74, 68]

for num in numbers:
    if num < 0:
        print(f"{num} is a negative number")
    else:
        print(f"{num} is a positive number")
```

This loop will go through each number in the list, assigning the number to the variable `num`, and then checking whether it's negative or not.



Loops allow us to execute a block of code multiple times. Here are two examples demonstrating the use of loops in Python.

3.2 Example 1: Sum of First N Natural Numbers using a for Loop

The following Python program calculates the sum of the first N natural numbers using a **for** loop.

3.2.1 Python Code

Listing 3: Sum of First N Natural Numbers

```
# Input: Number of terms
N = int(input("Enter a number: "))

# Initializing sum
total = 0

# Using a for loop to calculate sum
for i in range(1, N + 1):
    total += i

print("Sum of first", N, "natural numbers is:", total)
```

3.2.2 Explanation

- The user inputs a number N, which determines how many natural numbers to sum.
- A variable `total` is initialized to 0.
- The **for** loop iterates from 1 to N, adding each number to `total`.
- Finally, the total sum is printed.

3.3 Example 2: Finding Factorial using a while Loop

The following Python program calculates the factorial of a given number using a **while** loop.

3.3.1 Python Code

Listing 4: Factorial Calculation

```
# Input: Number for factorial
num = int(input("Enter a number: "))

# Initializing factorial
factorial = 1
```

```
6
7  # Using a while loop to calculate factorial
8  while num > 0:
9      factorial *= num
10     num -= 1
11
12 print("Factorial is:", factorial)
```

3.3.2 Explanation

- The user inputs a number whose factorial is to be calculated.
- A variable `factorial` is initialized to 1.
- The **while** loop runs until the number reduces to 0.
- In each iteration, the current number is multiplied with `factorial`, and the number is decremented.
- The final factorial value is printed.

4 Conclusion: Your Programming Journey Begins

We've covered a lot: making decisions with `if`, `elif`, and `else`; looping through lists with `for`; and visualizing code with Python Tutor. This is just the start of your programming journey. As you practice and explore, you'll learn more ways to create amazing programs that not only perform tasks but also think logically, just like the human mind.