Class #03: Operators and Conditional Statements:

- Introduction to operators and operands in Python.
- Arithmetic, comparison, and logical operators.
- Working with assignment, Boolean, and membership operators.
- Understanding if, if-else, and if-elif-else control flow statements.
- Nested if and nested if-else statements. **Assignment #3**: Conditional statements and operators practice.

আপনি **Python Operators** শিখতে চাইলে **W3Schools** একটি ভালো রিসোর্স হতে পারে। তাদের https://www.w3schools.com/python/python_operators.asp পেজে আপনি পাইথনের বিভিন্ন ধরনের অপারেটর (যেমন গাণিতিক অপারেটর, তুলনা অপারেটর, লজিক্যাল অপারেটর ইত্যাদি) সম্পর্কে বিস্তারিত জানতে পারবেন।

❖ Introduction to operators and operands in Python.

Python-এ **operators** এবং **operands** দুটি মৌলিক ধারণা, যা প্রোগ্রামিংয়ের জন্য অত্যন্ত গুরুত্বপূর্ণ।

1. Operands:

- **Operands** হলো তারা মান (values) বা ভ্যারিয়েবল (variables) যা কোনো অপারেশন-এ অংশগ্রহণ করে।
- সহজভাবে বলতে গেলে, **operand** হলো সেই সংখ্যাগুলো বা ডেটা যা **operator** এর মাধ্যমে গাণিতিক বা লজিক্যাল গাণনা সম্পন্ন করার জন্য ব্যবহৃত হয়।
- উদাহরণ: 5 + 3 এ, ৫ এবং ৩ হল **operands**।

সোজা কথায়, **operand** হলো চলক বা ভ্যারিয়েবল (variables) যেগুলোর উপর অপারেশন কার্যকর করা হয়।

2. Operators:

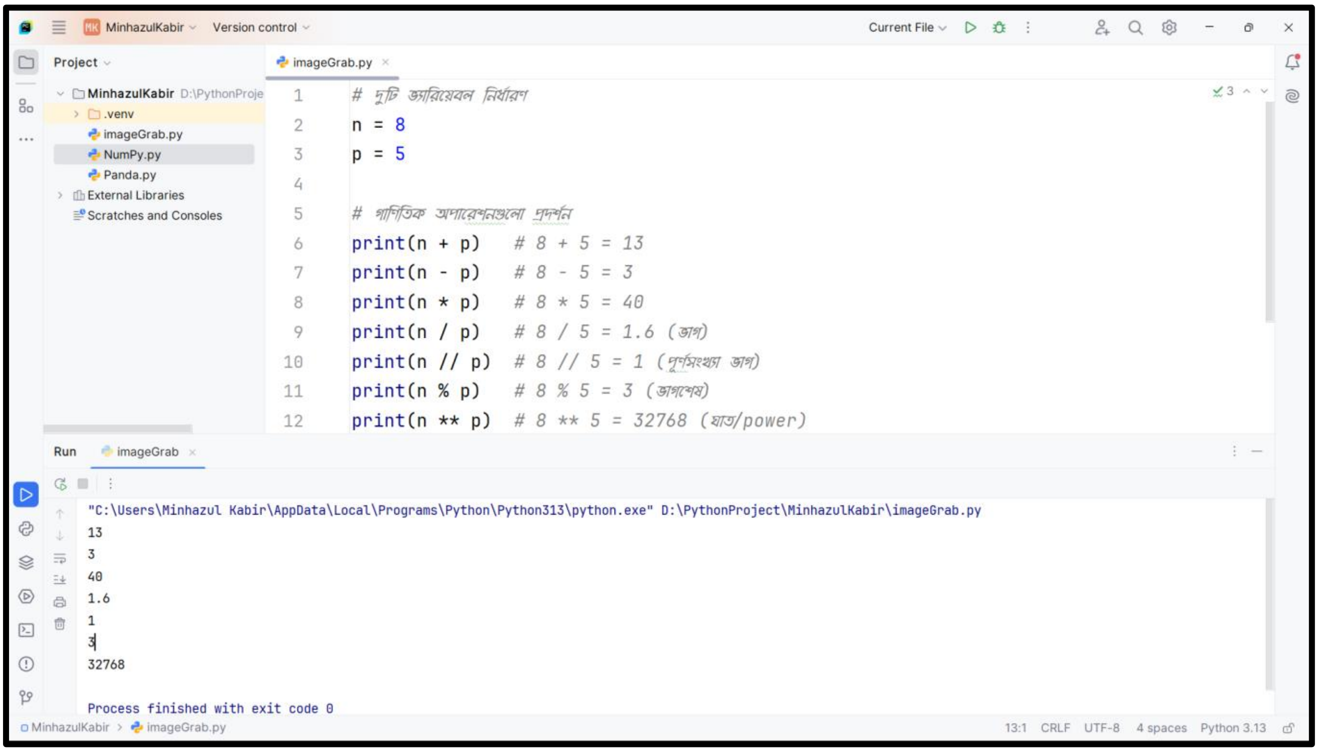
- **Operators** হলো সেগুলি যা **operands** (variables) এর মধ্যে কোনো কার্য সম্পাদন করতে ব্যবহৃত হয়।
- Python-এ বিভিন্ন ধরণের **operators** রয়েছে, যেমন: Arithmetic operators, Assignment operators, Comparison operators, Logical operators, Identity operators, Membership operators, Bitwise operators

❖ Arithmetic, comparison, and logical operators.

Python-এ Operators এর প্রকার:

1. Arithmetic Operators (গাণিতিক অপারেটর): এই অপারেটরগুলি গাণিতিক হিসাবের জন্য ব্যবহৃত হয়। এগুলি সংখ্যাগুলোর মধ্যে গাণিতিক অপারেশন সম্পাদন করে । গাণিতিক অপারেটরের কিছু উদাহরণ:

অপারেটর	নাম	ব্যাখ্যা	উদাহরণ
+	Addition: +	সংখ্যার যোগফল	5 + 3 = 8
-	Subtraction: -	সংখ্যার পার্থক্য	5 - 3 = 2
*	Multiplication: *	সংখ্যার গুণফল	5 * 3 = 15
/	Division: /	সংখ্যার ভাগফল (ফলাফল ভগ্নাংশ হবে)	5 / 3 = 1.66666666667
%	Modulus: %	ভাগশেষ	5 % 3 = 2
**	Exponentiation: ** (power)	একটি সংখ্যার নির্দিষ্ট ঘাত	5 ** 3 = 125 (5^3 = 125)
//	Floor Division: //	ভাগফলের ভগ্নাংশ অংশকে বাদ দেয়া	5 // 3 = 1



```python
# দুটি ভ্যারিয়েবল নির্ধারণ
n = 8
p = 5

# গাণিতিক অপারেশনগুলো প্রদর্শন
print(n + p)   # 8 + 5 = 13
print(n - p)   # 8 - 5 = 3
print(n * p)   # 8 * 5 = 40
print(n / p)   # 8 / 5 = 1.6 (ভাগ)
print(n // p)  # 8 // 5 = 1 (পূর্ণসংখ্যা ভাগ)
print(n % p)   # 8 % 5 = 3 (ভাগশেষ)
print(n ** p)  # 8 ** 5 = 32768 (ঘাত/power)
```

```
"C:\Users\Minhazul Kabir\AppData\Local\Programs\Python\Python313\python.exe" D:\PythonProject\MinhazulKabir\imageGrab.py
13
3
40
1.6
1
3
32768

Process finished with exit code 0
```

এখানে:

- n + p এর মান হবে 13 (যোগফল)
- n - p এর মান হবে 3 (বিয়োগফল)
- n * p এর মান হবে 40 (গুণফল)
- n / p এর মান হবে 1.6 (ভাগফল)
- n // p এর মান হবে 1 (পূর্ণসংখ্যা ভাগ) ভগ্নাংশ অংশ বাদ যাবে
- n % p এর মান হবে 3 (ভাগশেষ)
- n ** p এর মান হবে 32768 (ঘাত)

2. Comparison operators:

Python Comparison (Relational) Operators হলো সেগুলি অপারেটর যা দুটি মানের মধ্যে তুলনা করে এবং ফলস্বরূপ একটি **Boolean** মান (True বা False) প্রদান করে। এই অপারেটরগুলি মূলত শর্ত যাচাই (condition checking) করতে ব্যবহৃত হয়।

Comparison (Relational) Operators এর মান হতে পারে ২টাঃ

(i) True, Yes, 1
(ii) False, No, 0

Less than এবং **Greater than** হল **Relational Operators** (Comparison Operators) এর মধ্যে দুটি গুরুত্বপূর্ণ অপারেটর।

(i) n < 27 এখানে, x = 26, 25, 24, ...
(ii) n <= 27 এখানে, x = 27, 26, 25, 24, ...
(iii) n > 27 এখানে, x = 28, 29, 30, ...
(iv) n >= 27 এখানে, x = 27, 28, 29, 30, ...

নিচে **Python Comparison Operators** সম্পর্কিত ব্যাখ্যা দেওয়া হয়েছে যেখানে **n = 6, n = 27,** এবং **n = 100** এর মানের ভিত্তিতে প্রতিটি অপারেটরের ফলাফলগুলো একসাথে পর্যালোচনা করা হয়েছে।

Operator	**Description**	**যদি x = 6**	**যদি x = 27**	**যদি x = 100**
n == 27	== সমান কিনা (Equal to)	False	True	False
n != 27	!= অসমান কিনা (Not equal to)	True	False	True
n > 27	> বড় কিনা (Greater than)	False	False	True
n >= 27	>= বড় বা সমান কিনা (Greater than or equal to)	False	True	True
n < 27	< ছোট কিনা (Less than)	True	False	False
n <= 27	<= ছোট বা সমান কিনা (Less than or equal to)	True	True	False

১. যদি n = 6:

- **== (Equal to):** এই অপারেটরটি পরীক্ষা করে যে দুটি মান সমান কিনা। এখানে ৬ ২৭ এর সমান নয়, তাই ফলাফল হবে **False**।
- **!= (Not equal to):** এটি পরীক্ষা করে যে দুটি মান সমান নয় কি না। ৬ এবং ২৭ সমান নয়, তাই ফলাফল হবে **True**।
- **> (Greater than):** এই অপারেটরটি পরীক্ষা করে যে প্রথম মানটি দ্বিতীয় মানের চেয়ে বড় কিনা। ৬ ২৭ এর চেয়ে ছোট, তাই ফলাফল হবে **False**।
- **>= (Greater than or equal to):** এটি পরীক্ষা করে যে প্রথম মানটি দ্বিতীয় মানের চেয়ে বড় বা সমান কিনা। ৬ ২৭ এর থেকে ছোট, তাই ফলাফল হবে **False**।

- < (Less than): এটি পরীক্ষা করে যে প্রথম মানটি দ্বিতীয় মানের চেয়ে ছোট কিনা। ৬ ২৭ এর চেয়ে ছোট, তাই ফলাফল হবে True।
- <= (Less than or equal to): এটি পরীক্ষা করে যে প্রথম মানটি দ্বিতীয় মানের চেয়ে ছোট বা সমান কিনা। ৬ ২৭ এর চেয়ে ছোট, তাই ফলাফল হবে True।

২. যদি n = 27:

- == (Equal to): এখানে ২৭ ২৭ এর সমান, তাই ফলাফল হবে True।
- != (Not equal to): ২৭ ২৭ এর সমান, তাই ফলাফল হবে False।
- > (Greater than): ২৭ ২৭ এর চেয়ে বড় নয়, তাই ফলাফল হবে False।
- >= (Greater than or equal to): ২৭ ২৭ এর সমান, তাই ফলাফল হবে True।
- < (Less than): ২৭ ২৭ এর চেয়ে ছোট নয়, তাই ফলাফল হবে False।
- <= (Less than or equal to): ২৭ ২৭ এর সমান, তাই ফলাফল হবে True।

৩. যদি n = 100:

- == (Equal to): ১০০ ২৭ এর সমান নয়, তাই ফলাফল হবে False।
- != (Not equal to): ১০০ ২৭ এর সমান নয়, তাই ফলাফল হবে True।
- > (Greater than): ১০০ ২৭ এর থেকে বড়, তাই ফলাফল হবে True।
- >= (Greater than or equal to): ১০০ ২৭ এর থেকে বড়, তাই ফলাফল হবে True।
- < (Less than): ১০০ ২৭ এর থেকে ছোট নয়, তাই ফলাফল হবে False।
- <= (Less than or equal to): ১০০ ২৭ এর থেকে বড়, তাই ফলাফল হবে False।

```python
# n এর মান নির্ধারণ
n = 6

# তুলনা অপারেটরগুলো ব্যবহার করা
print("n == 27: ", n == 27)  # False, কারণ n 27 এর সমান নয়
print("n != 27: ", n != 27)  # True, কারণ n 27 এর সমান নয়
print("n > 27: ", n > 27)    # False, কারণ n 27 এর থেকে বড় নয়
print("n >= 27: ", n >= 27)  # False, কারণ n 27 এর সমান বা বড় নয়
print("n < 27: ", n < 27)    # True, কারণ n 27 এর থেকে ছোট
print("n <= 27: ", n <= 27)  # True, কারণ n 27 এর সমান বা ছোট
```

```
"C:\Users\Minhazul Kabir\AppData\Local\Programs\Python\Python313\python.exe" D:\PythonProject\MinhazulKabir\imageGrab.py
n == 27:  False
n != 27:  True
n > 27:  False
n >= 27:  False
n < 27:  True
n <= 27:  True

Process finished with exit code 0
```

3. **Logical operators:** **Logical/Boolean Operators** হলো সেই অপারেটর যা দুটি বা একাধিক শর্তের মধ্যে সম্পর্ক স্থাপন করে এবং ফলস্বরূপ **True** বা **False** প্রদান করে।

এগুলো সাধারণত **condition checking** বা **if-else** স্টেটমেন্টে ব্যবহৃত হয়, যেখানে আপনি নির্দিষ্ট শর্ত পূরণ কিনা পরীক্ষা করতে চান। তিনটি মৌলিক **Boolean Operators** হল:

যদি n=27 মান হয়।

1. **and Operator:** এটি দুটি শর্তের মধ্যে সম্পর্ক স্থাপন করে। যদি **উভয় শর্ত** সত্য (True) হয়, তবে ফলাফল হবে **True**, অন্যথায় **False**।

   উদাহরণ: n < 6 and n < 100 → এখানে, n 6 এর ছোট নয় (মিথ্যা), তাই ফলস্বরূপ **False** হবে। উভয় শর্ত সত্য নয়।

2. **or Operator:** এটি দুটি শর্তের মধ্যে সম্পর্ক স্থাপন করে। যদি **যেকোনো এক শর্ত** সত্য (True) হয়, তাহলে ফলাফল হবে **True**, অন্যথায় **False**।

   উদাহরণ: n < 6 or n < 100 → এখানে, n 100 এর ছোট (সত্য), তাই ফলস্বরূপ **True** হবে। যেকোনো একটা শর্ত সত্য ।

3. **not Operator:** এটি একক শর্তের বিপরীত মান প্রদান করে। যদি শর্তটি সত্য (True) হয়, তবে **False** এবং যদি মিথ্যা (False) হয়, তবে **True** হবে।

   উদাহরণ: not(n < 6 and n < 100) → এখানে, n < 6 and n < 100 ছিল **False**, তাই not(False) হবে **True**। False শর্তের বিপরীত মান True প্রদান করে।

```
# Variable declaration
n = 27   # এখানে n এর মান 27


# 1. and Operator
print("n < 6 and n < 100: ", n < 6 and n < 100)


# 2. or Operator
print("n < 6 or n < 100: ", n < 6 or n < 100)


# 3. not Operator
print("not(n < 6 and n < 100): ", not(n < 6 and n < 100))
```

```
n < 6 and n < 100:  False
n < 6 or n < 100:  True
not(n < 6 and n < 100):  True
```

❖ Working with assignment, Boolean, and membership operators.

Python Assignment Operators: হল সেই অপারেটরগুলো যা একটি মান ভেরিয়েবলে অ্যাসাইন বা নির্ধারণ করতে ব্যবহৃত হয়। Python-এ = চিহ্নটিকে **Assignment Operator** বলা হয়। এটি মূলত একটি মান বা এক্সপ্রেশনকে একটি ভেরিয়েবলে অ্যাসাইন (নির্ধারণ) করতে ব্যবহৃত হয়। এই অপারেটরটি ভেরিয়েবলের মান পরিবর্তন বা সেট করার জন্য ব্যবহৃত হয়। আমরা = কে 'সমান চিহ্ন' বলে থাকি ।

Operator	Description	Example
=	মান নির্ধারণ (Assign a value)	np = 5
+=	যোগফল নির্ধারণ (Add and assign)	np += 3 (অথবা np = np + 3)
-=	বিয়োগফল নির্ধারণ (Subtract and assign)	np -= 2 (অথবা np = np - 2)
*=	গুণফল নির্ধারণ (Multiply and assign)	np *= 4 (অথবা np = np * 4)
/=	ভাগফল নির্ধারণ (Divide and assign)	np /= 2 (অথবা np = np / 2)
//=	পূর্ণাঙ্গ ভাগফল নির্ধারণ (Floor divide and assign)	np //= 2 (অথবা np = np // 2)
%=	ভাগশেষ নির্ধারণ (Modulus and assign)	np %= 3 (অথবা np = np % 3)
**=	ঘাতফল নির্ধারণ (Exponent and assign)	np **= 2 (অথবা np = np ** 2)
&=	বিটওয়াইজ AND নির্ধারণ (Bitwise AND and assign)	np &= 2 (অথবা np = np & 2)
`	=`	বিটওয়াইজ OR নির্ধারণ (Bitwise OR and assign)
^=	বিটওয়াইজ XOR নির্ধারণ (Bitwise XOR and assign)	np ^= 4 (অথবা np = np ^ 4)
<<=	বিটওয়াইজ left shift নির্ধারণ (Bitwise left shift and assign)	np <<= 2 (অথবা np = np << 2)
>>=	বিটওয়াইজ right shift নির্ধারণ (Bitwise right shift and assign)	np >>= 2 (অথবা np = np >> 2)

```
# Variable declaration
np = 5  # Initially, np is 5

# 1. Add and assign
np += 3  # Equivalent to np = np + 3
print("np += 3:", np)  # Output: np = 8

# 2. Subtract and assign
np -= 2  # Equivalent to np = np - 2
print("np -= 2:", np)  # Output: np = 6
```

```
# 3. Multiply and assign
np *= 4  # Equivalent to np = np * 4
print("np *= 4:", np)  # Output: np = 24

# 4. Divide and assign
np /= 2  # Equivalent to np = np / 2
print("np /= 2:", np)  # Output: np = 12.0 (float result)

# 5. Floor divide and assign
np //= 2  # Equivalent to np = np // 2
print("np //= 2:", np)  # Output: np = 6 (floor division)

# 6. Modulus and assign
np %= 3  # Equivalent to np = np % 3
print("np %= 3:", np)  # Output: np = 0 (remainder of 6 divided by
3)

# 7. Exponent and assign
np **= 2  # Equivalent to np = np ** 2
print("np **= 2:", np)  # Output: np = 0 (0^2 = 0)

# 8. Bitwise AND and assign
np = 5  # Reset np to 5, binary value of 5 is 101
np &= 2  # binary value of 2 is 010, bitwise AND of 5 and 2 is 000
print("np &= 2:", np)  # Output: np = 0

# 9. Bitwise OR and assign
np |= 3  # binary value of 3 is 011, bitwise OR of 0 and 3 is 011
print("np |= 3:", np)  # Output: np = 3

# 10. Bitwise XOR and assign
np ^= 4  # binary value of 4 is 100, bitwise XOR of 3 and 4 is 7
print("np ^= 4:", np)  # Output: np = 7

# 11. Bitwise left shift and assign
np <<= 2  # Left shift 7 (binary 0111) by 2 bits, gives 28 (binary
11100)
print("np <<= 2:", np)  # Output: np = 28

# 12. Bitwise right shift and assign
np >>= 2  # Right shift 28 (binary 11100) by 2 bits, gives 7
(binary 0111)
print("np >>= 2:", np)  # Output: np = 7
```

```
np += 3: 8
np -= 2: 6
np *= 4: 24
np /= 2: 12.0
np //= 2: 6.0
np %= 3: 0.0
np **= 2: 0.0
np &= 2: 0
np |= 3: 3
```

```
np ^= 4: 7
np <<= 2: 28
np >>= 2: 7
```

প্রথমে, np = 5 দিয়ে ভেরিয়েবলের মান ৫ সেট করা হয়েছে।

1. np += 3 অপারেটরটি মানে np এর বর্তমান মানে ৩ যোগ করা। প্রথমে ৫ ছিল, তাই ৫ + ৩ = ৮ হয়ে যায়।
2. np -= 2 অপারেটরটি মানে np এর বর্তমান মান থেকে ২ বিয়োগ করা। ৮ থেকে ২ বিয়োগ করলে ৬ হয়।
3. np *= 4 অপারেটরটি মানে np এর বর্তমান মানে ৪ গুণ করা। ৬ গুণ ৪ হলে ২৪ হয়।
4. np /= 2 অপারেটরটি মানে np এর বর্তমান মান ২ দ্বারা ভাগ করা। ২৪ ভাগ ২ হলে ১২.০ (ফ্লোট) হয়।
5. np //= 2 অপারেটরটি মানে np এর বর্তমান মানকে ২ দিয়ে পূর্ণাঙ্গভাবে ভাগ করা। ১২.০ পূর্ণাঙ্গভাবে ২ দ্বারা ভাগ করলে ৬ হয়।
6. np %= 3 অপারেটরটি মানে np এর বর্তমান মান ৩ দিয়ে ভাগশেষ নেয়া। ৬ এর ভাগশেষ ৩ দিয়ে ০ হবে।
7. np **= 2 অপারেটরটি মানে np এর বর্তমান মানের ঘাতফল নেয়া। ০ এর ঘাতফল ০ হওয়ায় np = 0।
8. np = 5 দিয়ে আবার np এর মান ৫ সেট করা হলো এবং np &= 2 অপারেটরটি মানে বিটওয়াইজ AND অপারেশন। ৫ (বাইনারি 101) এবং ২ (বাইনারি 010) এর বিটওয়াইজ AND এর ফলাফল ০ (বাইনারি 000) হবে।
9. np |= 3 অপারেটরটি বিটওয়াইজ OR অপারেশন। ০ (বাইনারি 000) এবং ৩ (বাইনারি 011) এর বিটওয়াইজ OR এর ফলাফল ৩ (বাইনারি 011) হবে।
10. np ^= 4 অপারেটরটি বিটওয়াইজ XOR অপারেশন। ৩ (বাইনারি 011) এবং ৪ (বাইনারি 100) এর বিটওয়াইজ XOR এর ফলাফল ৭ (বাইনারি 111) হবে।
11. np <<= 2 অপারেটরটি বিটওয়াইজ বাম শিফট অপারেশন। ৭ (বাইনারি 0111) বাম দিকে ২ বিট শিফট করলে ২৮ (বাইনারি 11100) হবে।
12. np >>= 2 অপারেটরটি বিটওয়াইজ ডান শিফট অপারেশন। ২৮ (বাইনারি 11100) ডান দিকে ২ বিট শিফট করলে ৭ (বাইনারি 0111) হবে।

Python Membership Operators হলো দুটি অপারেটর, যা সিকোয়েন্সের মধ্যে কোনো উপাদান উপস্থিত আছে কিনা তা চেক করতে ব্যবহৃত হয়:

1. **in:** এই অপারেটরটি চেক করে যে একটি নির্দিষ্ট উপাদান সিকোয়েন্সে (যেমন লিস্ট, টিউপল, স্ট্রিং, সেট ইত্যাদি) উপস্থিত আছে কিনা। যদি উপাদান সিকোয়েন্সে থাকে, তাহলে এটি True রিটার্ন করে, আর যদি না থাকে, তাহলে False রিটার্ন করে।
2. **not in:** এই অপারেটরটি চেক করে যে একটি নির্দিষ্ট উপাদান সিকোয়েন্সে উপস্থিত নেই। যদি উপাদান সিকোয়েন্সে না থাকে, তাহলে এটি True রিটার্ন করে, আর যদি থাকে, তাহলে False রিটার্ন করে।

এই অপারেটরগুলি সাধারণত লিস্ট, টিউপল, স্ট্রিং, সেট এবং ডিকশনারি এর মধ্যে উপাদান উপস্থিতি যাচাই করতে ব্যবহৃত হয়।

❖ Understanding if, if-else, and if-elif-else control flow statements.

if, elif, else এর গঠনঃ

```
if condition1 (Comparison/Relational operator):
    # Code to execute if condition1 is True
elif condition2 (Comparison/Relational operator):
    # Code to execute if condition1 is False and condition2 is True
elif condition3 (Comparison/Relational operator):
    # Code to execute if condition1 and condition2 are False, but
condition3 is True
elif condition4 (Comparison/Relational operator):
    # Code to execute if condition1, condition2, and condition3 are
False, but condition4 is True
elif condition5 (Comparison/Relational operator):
    # Code to execute if condition1, condition2, condition3, and
condition4 are False, but condition5 is True
else:
    # Code to execute if all conditions (condition1, condition2,
condition3, condition4, condition5) are False
```

1) if দিয়ে শুরু হবে এবং else দিয়ে শেষ হবে । if এবং else এর মাঝে অসংখ্য (infinity) elif থাকতে পারে ।
2) else লিখতে হলে আগে বাধ্যতামূলক if লিখতে হবে ।
3) যতগুলো if বা elif থাকবে, ঠিক ততগুলোই condition1 অর্থাৎ (Comparison/Relational operator) হবে ।
4) else এর পরে কোনো relational operator হয় না।
5) if বা elif এ কোলনের মধ্যে Comparison/Relational operator হবে।
    - ✓ Comparison/Relational operator এর মান সত্য হলে সে statement কাজ করবে ।
    - ✓ Comparison/Relational operator এর মান সত্য না হলে, পরবর্তি ধাপে যাবে।
6) if, elif, else এর পরে বাধ্যতামুলক কোলন (:) লিখতে হবে। এবং পরের লাইনে indentation (স্পেস বা ট্যাব) ব্যবহার না করলে পাইথন সঠিকভাবে কোডটি বুঝতে পারবে না এবং সঠিকভাবে কাজ করবে না।

"পাইথনে একটি সংখ্যা ধনাত্মক, ঋণাত্মক, বা শূন্য কিনা তা চেক করার কোড এবং ব্যাখ্যা"

```
# Taking input from the user
number = float(input("Enter a number: "))

# Checking the number
if number > 0:
    print("The number is positive.")
elif number < 0:
    print("The number is negative.")
else:
    print("The number is zero.")
```

ব্যাখ্যা:

1. **ব্যবহারকারীর কাছ থেকে ইনপুট নেয়া:**
    - float(input("Enter a number: "))
      এখানে input() ফাংশনটি ব্যবহার করে ব্যবহারকারীর কাছ থেকে একটি সংখ্যা ইনপুট নেওয়া হচ্ছে।
    - float() ফাংশনটি ব্যবহার করা হয়েছে যাতে ইনপুটটি একটি দশমিক সংখ্যা (যেমন 5.5) হতে পারে, না হলে এটি শুধুমাত্র পূর্ণসংখ্যা হিসেবে নেওয়া হত।
2. **সংখ্যা চেক করা:**
    - if number > 0: প্রথমে if স্টেটমেন্টটি চেক করে যে সংখ্যা ধনাত্মক কি না (যেমন, 5, 10, 20)।
        - যদি সংখ্যাটি ধনাত্মক হয়, তাহলে " The number is positive." এই বার্তা প্রিন্ট হবে।
    - elif number < 0: যদি প্রথম শর্ত মিথ্যা হয় (অর্থাৎ সংখ্যা ধনাত্মক না হয়), তাহলে elif স্টেটমেন্টটি চেক করবে যে সংখ্যা ঋণাত্মক কি না (যেমন -5, -10, -20)।
        - যদি সংখ্যাটি ঋণাত্মক হয়, তাহলে " The number is negative." এই বার্তা প্রিন্ট হবে।
    - else: যদি সংখ্যাটি না ধনাত্মক হয়, না ঋণাত্মক হয়, তাহলে এটি শূন্য হবে (যেমন 0)।
        - শূন্য হলে " The number is zero." এই বার্তা প্রিন্ট হবে।

এভাবে কোডটি ব্যবহারকারীর ইনপুট অনুযায়ী নির্ধারণ করে দেয় যে সংখ্যা ধনাত্মক, ঋণাত্মক, নাকি শূন্য।

প্রশ্ন: একটি প্রোগ্রাম লিখুন যা ব্যবহারকারীর GPA (Grade Point Average) ইনপুট নেবে এবং সেই GPA এর মান অনুযায়ী গ্রেড নির্ধারণ করবে। নিম্নলিখিত শর্তাবলী অনুযায়ী গ্রেড নির্ধারণ করতে হবে:

- GPA যদি 3.7 বা তার বেশি হয়, তাহলে গ্রেড হবে A।
- GPA যদি 3.0 বা তার বেশি, কিন্তু 3.7 এর কম হয়, তাহলে গ্রেড হবে B।
- GPA যদি 2.0 বা তার বেশি, কিন্তু 3.0 এর কম হয়, তাহলে গ্রেড হবে C।
- GPA যদি 1.0 বা তার বেশি, কিন্তু 2.0 এর কম হয়, তাহলে গ্রেড হবে D।
- GPA যদি 1.0 এর কম হয়, তাহলে গ্রেড হবে F।

প্রোগ্রামটি ব্যবহারকারীর GPA ইনপুট নিয়ে সঠিক গ্রেড প্রদর্শন করবে।

```
# Taking GPA input from the user
gpa = float(input("Enter your GPA (0.0 to 4.0): "))

# Checking the GPA and assigning grades
if gpa >= 3.7:
    print("Your grade is A")
elif gpa >= 3.0:
    print("Your grade is B")
elif gpa >= 2.0:
    print("Your grade is C")
```

```
elif gpa >= 1.0:
    print("Your grade is D")
else:
    print("Your grade is F")
```

নির্দিষ্টভাবে ব্যাখ্যা করা হলে, প্রোগ্রামটি ব্যবহারকারীর GPA ইনপুট নেয় এবং তার ভিত্তিতে একটি গ্রেড প্রদর্শন করে। এখানে কীভাবে এটি কাজ করে:

1. **GPA ইনপুট**: প্রথমে ব্যবহারকারীকে তাদের GPA (Grade Point Average) দিতে বলা হয়, যা 0.0 থেকে 4.0 এর মধ্যে হতে হবে।
2. **গ্রেড নির্ধারণ**:
   - যদি GPA 3.7 বা তার বেশি হয়, তবে গ্রেড হবে **A**।
   - যদি GPA 3.0 বা তার বেশি, কিন্তু 3.7 এর কম হয়, তাহলে গ্রেড হবে **B**।
   - যদি GPA 2.0 বা তার বেশি, কিন্তু 3.0 এর কম হয়, তখন গ্রেড হবে **C**।
   - যদি GPA 1.0 বা তার বেশি, কিন্তু 2.0 এর কম হয়, তাহলে গ্রেড হবে **D**।
   - যদি GPA 1.0 এর কম হয়, তাহলে গ্রেড হবে **F**।

এই শর্তগুলো পরীক্ষা করে, প্রোগ্রামটি গ্রেডের সাথে মেলে এমন আউটপুট প্রদর্শন করবে।

উদাহরণ:

- যদি ব্যবহারকারী GPA হিসেবে 3.8 প্রদান করে, তাহলে আউটপুট হবে **A**।
- যদি GPA 2.5 হয়, তাহলে আউটপুট হবে **C**।
- যদি GPA 0.5 হয়, তখন আউটপুট হবে **F**।

প্রোগ্রামটি GPA এর মান অনুযায়ী সঠিক গ্রেড ঘোষণা করবে।

❖ Nested if and nested if-else statements

Nest এর বাংলা অর্থ পাখির বাসা। একটা if, if-else এর মধ্যে নতুনভাবে if, if-else লিখলে তাকে nested বলে

সারাংশ:

- **Nested if**: এক if এর ভিতরে আরেকটি if থাকে, যেখানে একটি শর্তের জন্য অন্য শর্তের উপর ভিত্তি করে সিদ্ধান্ত নেওয়া হয়।
- **Nested if-else**: এক if এর ভিতরে if-else থাকে, যেখানে শর্ত অনুযায়ী দুটি আলাদা পথ (যেমন, if এবং else) থেকে নির্বাচন করা হয়।

এগুলোকে সাধারণত ব্যবহার করা হয় যখন একাধিক শর্ত এবং তাদের ফলাফল পর্যালোচনা করতে হয়।

প্রশ্ন: একটি প্রোগ্রাম লিখুন যা তিনটি সংখ্যার মধ্যে বৃহত্তম সংখ্যা নির্ধারণ করবে। প্রোগ্রামটি **nested if-else** ব্যবহার করে তিনটি সংখ্যার মধ্যে বড়টি খুঁজে বের করবে এবং সেই সংখ্যা প্রদর্শন করবে।

```
# তিনটি সংখ্যা ইনপুট নেওয়া
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
num3 = float(input("Enter the third number: "))

# Nested if-else দিয়ে বড় সংখ্যা নির্ধারণ করা
if num1 >= num2:
    if num1 >= num3:
        print(f"The largest number is {num1}.")
    else:
        print(f"The largest number is {num3}.")
else:
    if num2 >= num3:
        print(f"The largest number is {num2}.")
    else:
        print(f"The largest number is {num3}.")
```

উদাহরণ:

1. **যদি ইনপুট দেওয়া হয়:**
    - num1 = 12, num2 = 7, num3 = 9
    - আউটপুট হবে: The largest number is 12.
2. **যদি ইনপুট দেওয়া হয়:**
    - num1 = 5, num2 = 8, num3 = 3
    - আউটপুট হবে: The largest number is 8.
3. **যদি ইনপুট দেওয়া হয়:**
    - num1 = 7, num2 = 9, num3 = 9
    - আউটপুট হবে: The largest number is 9.

এভাবে, nested if-else ব্যবহার করে তিনটি সংখ্যার মধ্যে বৃহত্তম সংখ্যা নির্ধারণ করা হয়েছে।

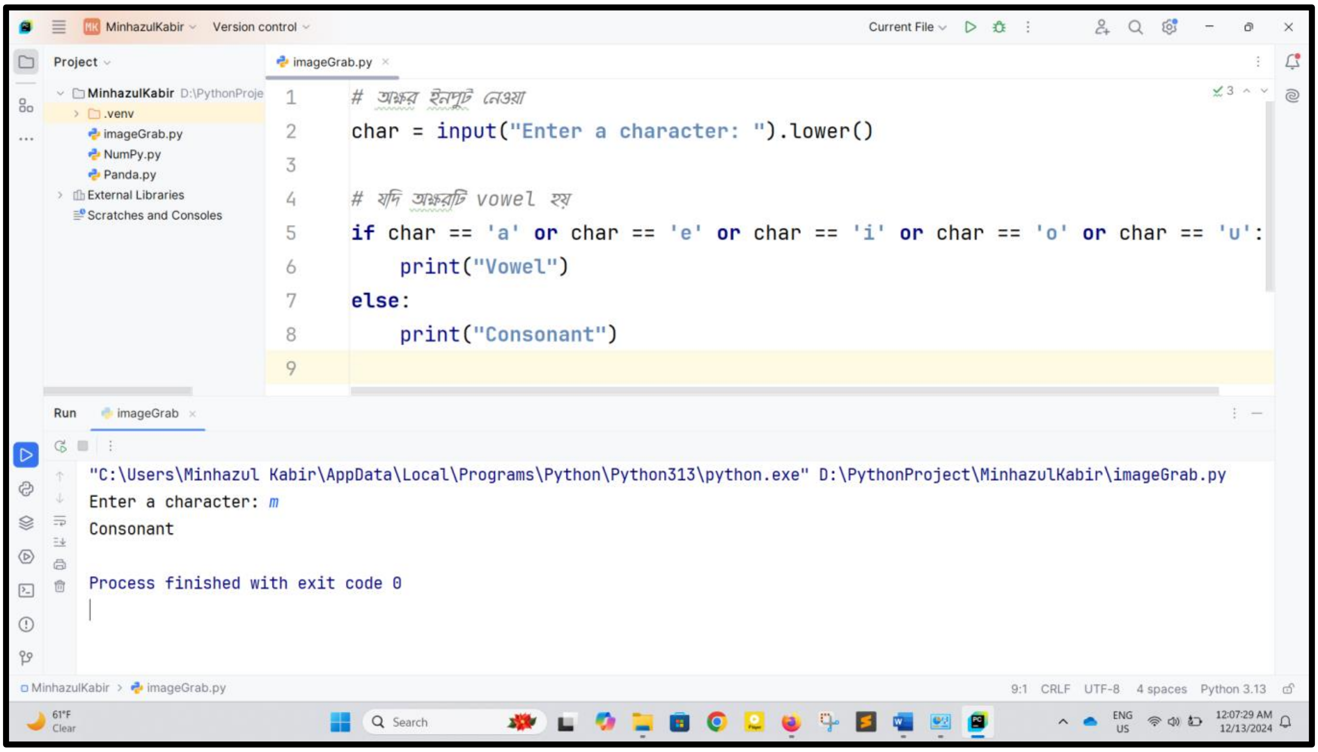
প্রশ্ন: একটি প্রোগ্রাম লিখুন যা একটি অক্ষর ইনপুট নেবে এবং সেই অক্ষরটি **vowel** (স্বল্পবর্ণ) কিনা তা যাচাই করবে। যদি অক্ষরটি **vowel** হয়, তবে "Vowel" প্রদর্শন করবে, অন্যথায় "Consonant" প্রদর্শন করবে।

উত্তর:

এখানে if এবং else ব্যবহার করে প্রোগ্রামটি কীভাবে লেখা হবে তা ব্যাখ্যা করা হলো:

ব্যাখ্যা:

- **vowel** অক্ষরগুলি হল a, e, i, o, u (যা বড় হাতেও হতে পারে, যেমন A, E, I, O, U)।
- যদি ইনপুট অক্ষরটি এসবের মধ্যে একটি হয়, তবে এটি **vowel**, অন্যথায় **consonant**।



```python
# অক্ষর ইনপুট নেওয়া
char = input("Enter a character: ").lower()

# যদি অক্ষরটি vowel হয়
if char == 'a' or char == 'e' or char == 'i' or char == 'o' or char == 'u':
    print("Vowel")
else:
    print("Consonant")
```

```
"C:\Users\Minhazul Kabir\AppData\Local\Programs\Python\Python313\python.exe" D:\PythonProject\MinhazulKabir\imageGrab.py
Enter a character: m
Consonant

Process finished with exit code 0
```

এই প্রোগ্রামটি একটি অক্ষর ইনপুট নেয় এবং সেই অক্ষরটি **vowel** (স্বল্পবর্ণ) কিনা তা যাচাই করে। যদি অক্ষরটি ভাওয়েল হয়, তবে "Vowel" প্রিন্ট হবে, আর যদি কনসোন্যান্ট হয়, তবে "Consonant" প্রিন্ট হবে। চলুন, প্রোগ্রামটির প্রতিটি অংশ কীভাবে কাজ করে, তা বিস্তারিতভাবে ব্যাখ্যা করি:

1. অক্ষর ইনপুট নেওয়া:

প্রথমে ব্যবহারকারীর কাছ থেকে একটি অক্ষর ইনপুট নেওয়া হয়:

char = input("Enter a character: ").lower()

- input() ফাংশন ব্যবহার করে ব্যবহারকারী একটি অক্ষর ইনপুট দেয়। ব্যবহারকারী যে অক্ষরটি ইনপুট দেয় তা একটি স্ট্রিং হিসেবে গ্রহণ করা হয়।
- .lower() মেথডটি ব্যবহার করা হয়েছে যাতে ইনপুটted অক্ষরটি যে কোনও আকারে (বড় বা ছোট হাতের) আসুক, সেটা সব সময় ছোট হাতের (lowercase) আকারে রূপান্তরিত হয়। উদাহরণস্বরূপ, যদি ব্যবহারকারী

A ইনপুট দেয়, তাহলে .lower() সেটিকে a তে রূপান্তরিত করবে। এর ফলে, ভাওয়েল চেক করার সময় বড় হাতের অক্ষর এবং ছোট হাতের অক্ষরের জন্য আলাদা আলাদা শর্ত তৈরি করার প্রয়োজন নেই।

2. if-else শর্ত দিয়ে vowel বা consonant চেক করা:

এবার, if শর্তের মাধ্যমে চেক করা হচ্ছে যে ইনপুটted অক্ষরটি **vowel** (যেমন a, e, i, o, u) কিনা:

```
if char == 'a' or char == 'e' or char == 'i' or char == 'o' or char == 'u':
    print("Vowel")
else:
    print("Consonant")
```

- **if শর্ত**: এই অংশে, char এর মান যদি a, e, i, o, বা u হয়, তাহলে শর্তটি সত্য হবে এবং প্রোগ্রামটি "Vowel" প্রিন্ট করবে।
- এখানে or অপারেটরটি ব্যবহার করা হয়েছে। or অপারেটরটি একাধিক শর্ত পরীক্ষা করতে ব্যবহার করা হয়। অর্থাৎ, যদি ইনপুটted অক্ষরটি যেকোনো একটির সাথে মিলে যায়, তাহলে শর্তটি সত্য হবে।
- **else শর্ত**: যদি if শর্তটি মিথ্যা হয় (অর্থাৎ, ইনপুটted অক্ষরটি ভাওয়েল না হয়), তবে else ব্লকটি কার্যকর হবে এবং প্রোগ্রামটি "Consonant" প্রিন্ট করবে।

উদাহরণ:

1. **যদি ইনপুট অক্ষর হয়** a:
    - .lower() মেথডের মাধ্যমে এটি ছোট হাতের a তে রূপান্তরিত হবে।
    - if শর্তটি চেক করবে, এবং যেহেতু a ভাওয়েল, আউটপুট হবে: **"Vowel"**।
2. **যদি ইনপুট অক্ষর হয়** B:
    - .lower() মেথডের মাধ্যমে এটি ছোট হাতের b তে রূপান্তরিত হবে।
    - if শর্তটি মিথ্যা হবে (কারণ b ভাওয়েল নয়), এবং else ব্লকটি কার্যকর হবে, আউটপুট হবে: **"Consonant"**।

Pass:

pass একটি বিশেষ কিবোর্ড শব্দ বা স্টেটমেন্ট যা পাইথনে **কিছুই না করার** জন্য ব্যবহৃত হয়। এটি প্রোগ্রামের লজিকের মধ্যে কোনো নির্দিষ্ট অংশে কিছু না করার উদ্দেশ্যে রাখা হয়, যেখানে সিনট্যাক্সগতভাবে কিছু করার দরকার কিন্তু কার্যকরী কিছু করার প্রয়োজন নেই।

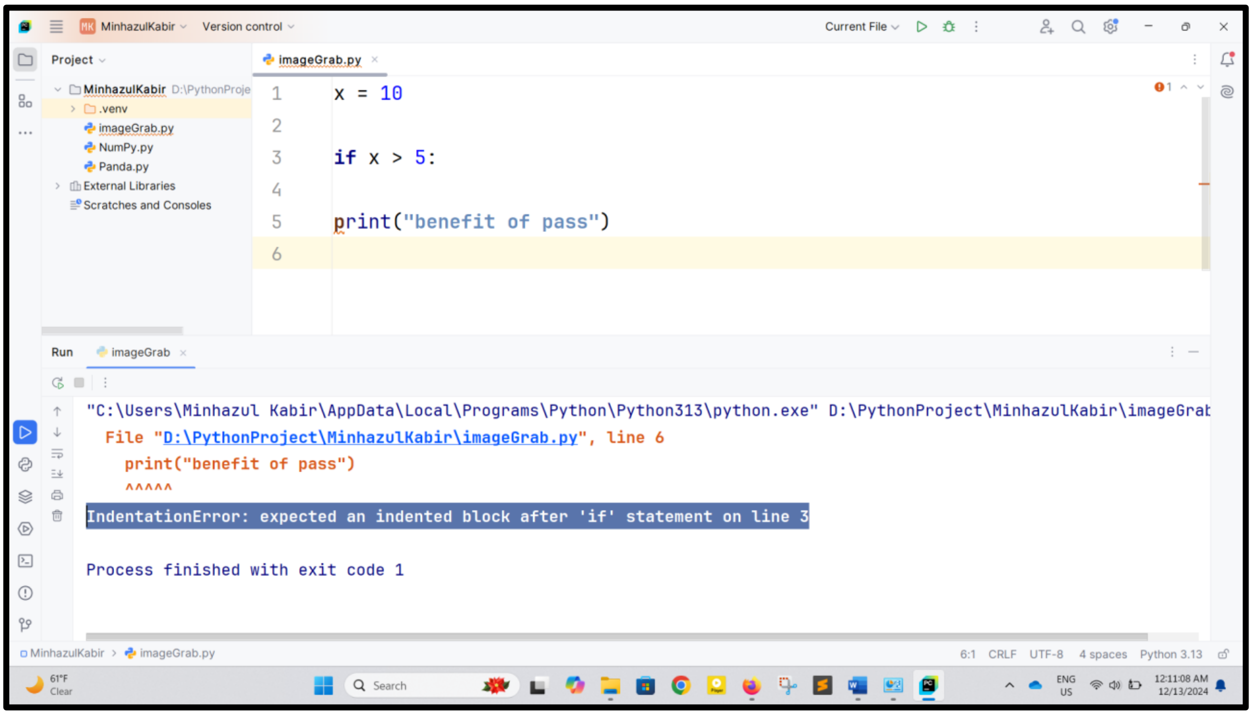
pass ব্যবহার কেনো করা হয়?

1. **স্টাব বা প্লেসহোল্ডার হিসেবে:**
    - যখন আপনি কোনো ফাংশন, ক্লাস, লুপ, বা শর্তের জন্য স্ট্রাকচার তৈরি করছেন কিন্তু পরে তার ভিতরে কাজ করবেন, তখন pass ব্যবহার করতে পারেন। এটি সিনট্যাক্সের ত্রুটি এড়াতে সাহায্য করে, যেমন একটি ফাংশন বা শর্তে অন্তত একটি statement থাকা আবশ্যক।
2. **কোনো কোড ব্লক খালি রাখার জন্য:**

- কিছু ক্ষেত্রে কোড ব্লকটি খালি রাখার দরকার হতে পারে, যেমন একটি নির্দিষ্ট শর্তে কোনো কাজ না করা, তবে আপনি ভুলবশত ত্রুটি ফেলতে চান না, তখন pass ব্যবহার করতে পারেন।

3. **অপর্যাপ্ত কোডের জন্য:**
   - যখন আপনি কোড লেখার সময় কিছু অংশ পরে তৈরি করতে চান, তখন সেই জায়গায় pass রেখে দেবেন যাতে প্রোগ্রামটি চলতে থাকে।

Without pass: IndentationError: expected an indented block after 'if' statement on line 3

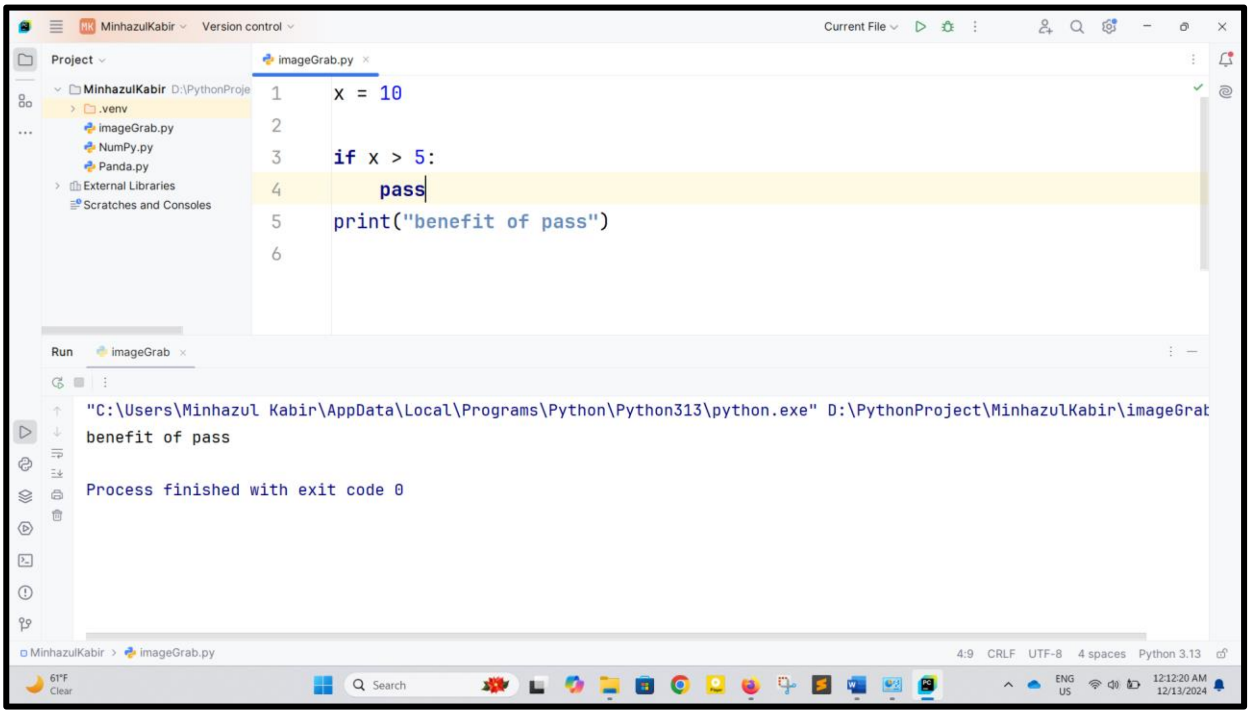


```
1   x = 10
2
3   if x > 5:
4
5   print("benefit of pass")
6
```

```
"C:\Users\Minhazul Kabir\AppData\Local\Programs\Python\Python313\python.exe" D:\PythonProject\MinhazulKabir\imageGrab
  File "D:\PythonProject\MinhazulKabir\imageGrab.py", line 6
    print("benefit of pass")
    ^^^^^
IndentationError: expected an indented block after 'if' statement on line 3

Process finished with exit code 1
```

pass:



```
1   x = 10
2
3   if x > 5:
4       pass
5   print("benefit of pass")
6
```

```
"C:\Users\Minhazul Kabir\AppData\Local\Programs\Python\Python313\python.exe" D:\PythonProject\MinhazulKabir\imageGrab
benefit of pass

Process finished with exit code 0
```