| age = 20  if age >= 18:  print("You are eligible to vote.")  else:  print("You are not eligible to vote.") |
| --- |

| score = 85  if score >= 90:  print("Grade: A")  elif score >= 75:  print("Grade: B")  else:  print("Grade: C") |
| --- |

| x = 10  y = 5  if x > 0:  if y > 0:  print("Both x and y are positive.")  else:  print("x is positive but y is not.")  else:  print("x is not positive.") |
| --- |

| fruits = ["apple", "banana", "cherry"]    for fruit in fruits:  print(fruit) |
| --- |

| number = 5  factorial = 1    for i in range(1, number + 1):  factorial \*= i    print("Factorial of", number, "is", factorial) |
| --- |

| print("Even numbers from 0 to 20:")  for i in range(0, 21, 2):  print(i) |
| --- |

| matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]    print("Matrix elements:")  for row in matrix:  for element in row:  print(element, end=' ')  print() # for new line after each row |
| --- |

| i = 1  while i <= 5:  print(i)  i += 1 |
| --- |

| number = None  while number is None or number < 0:  try:  number = int(input("Enter a positive number: "))  if number < 0:  print("Invalid input. Please enter a positive number.")  except ValueError:  print("That's not a number. Try again.") |
| --- |

| for \_ in range(3):  try:  number = int(input("Enter a positive number: "))  if number > 0:  print(f"Valid number entered: {number}")  break  else:  print("Invalid input. Please enter a positive number.")  except ValueError:  print("That's not a number. Try again.")  else:  print("Sorry, you've used all attempts.") |
| --- |

| int number;  do {  printf("Enter a positive number: ");  scanf("%d", &number);  } while (number <= 0); |
| --- |

| int main() {  int x = 10;  int y = 0;  if (x > 5) {  y = x - 5;  } else {  y = x + 5;  }  return y;  } |
| --- |

| init: // Initialization  x = 0  condition:  if x < n: jump to body  jump to end  body:  // Execute loop statements  // ...  x = x + 1 // Increment  jump condition  end: |
| --- |

| Example of Intermediate Code:  explain in simple points for both for and while loop  condition:  if condition: jump to body  jump to end  body:  // Execute loop statements  // ...  jump condition  end: |
| --- |

| Example of Intermediate Code:  explain in simple points for both for and while loop  body:  // Execute loop statements  // ...  condition:  if condition: jump body  end: |
| --- |

| define i32 @add(i32 %a, i32 %b) {  %sum = add i32 %a, %b  ret i32 %sum  } |
| --- |

| int sum = 0;  for (int i = 1; i <= 10; i++) {  sum += i;  } |
| --- |

| %sum = alloca i32  store i32 0, i32\* %sum  %i = alloca i32  store i32 1, i32\* %i  br label %loop\_condition  loop\_condition:  %i\_val = load i32, i32\* %i  %cond = icmp le i32 %i\_val, 10  br i1 %cond, label %loop\_body, label %loop\_end  loop\_body:  %current\_sum = load i32, i32\* %sum  %i\_val = load i32, i32\* %i  %new\_sum = add i32 %current\_sum, %i\_val  store i32 %new\_sum, i32\* %sum  %next\_i = add i32 %i\_val, 1  store i32 %next\_i, i32\* %i  br label %loop\_condition  loop\_end: |
| --- |

| int factorial = 1;  int n = 5; // Assume n is initialized  while (n > 1) {  factorial \*= n;  n--;  } |
| --- |

| %factorial = alloca i32  store i32 1, i32\* %factorial  %n = alloca i32  store i32 5, i32\* %n  br label %while\_condition  while\_condition:  %n\_val = load i32, i32\* %n  %cond = icmp gt i32 %n\_val, 1  br i1 %cond, label %while\_body, label %while\_end  while\_body:  %current\_factorial = load i32, i32\* %factorial  %n\_val = load i32, i32\* %n  %new\_factorial = mul i32 %current\_factorial, %n\_val  store i32 %new\_factorial, i32\* %factorial  %decremented\_n = sub i32 %n\_val, 1  store i32 %decremented\_n, i32\* %n  br label %while\_condition  while\_end: |
| --- |

| int n = 1;  do {  printf("%d\n", n);  n++;  } while (n <= 5); |
| --- |

| %n = alloca i32  store i32 1, i32\* %n  br label %do\_while\_body  do\_while\_body:  %n\_val = load i32, i32\* %n  call void @printf(i32 %n\_val) // Assume printf is declared appropriately  %incremented\_n = add i32 %n\_val, 1  store i32 %incremented\_n, i32\* %n  %cond\_check = icmp le i32 %incremented\_n, 5  br i1 %cond\_check, label %do\_while\_body, label %do\_while\_end  do\_while\_end: |
| --- |

| int x = 10;  int result;  if (x > 5) {  result = x \* 2;  } else {  result = x + 2;  } |
| --- |

| %x = alloca i32  store i32 10, i32\* %x  %result = alloca i32  %x\_val = load i32, i32\* %x  %cond = icmp gt i32 %x\_val, 5  br i1 %cond, label %if\_true, label %if\_false  if\_true:  %true\_value = mul i32 %x\_val, 2  store i32 %true\_value, i32\* %result  br label %if\_end  if\_false:  %false\_value = add i32 %x\_val, 2  store i32 %false\_value, i32\* %result  br label %if\_end  if\_end: |
| --- |

| int case\_num = 2;  switch (case\_num) {  case 1:  printf("One");  break;  case 2:  printf("Two");  break;  default:  printf("Default");  } |
| --- |

| %case\_num = alloca i32  store i32 2, i32\* %case\_num  %case\_val = load i32, i32\* %case\_num  switch i32 %case\_val, label %default\_case [  i32 1, label %case\_one  i32 2, label %case\_two  ]  case\_one:  call void @printf(i8\* getelementptr inbounds ([4 x i8], [4 x i8]\* @str1, i32 0, i32 0)) // "One"  br label %end\_switch  case\_two:  call void @printf(i8\* getelementptr inbounds ([4 x i8], [4 x i8]\* @str2, i32 0, i32 0)) // "Two"  br label %end\_switch  default\_case:  call void @printf(i8\* getelementptr inbounds ([7 x i8], [7 x i8]\* @str3, i32 0, i32 0)) // "Default"  br label %end\_switch  end\_switch: |
| --- |

| int add(int a, int b) {  return a + b;  }  int result = add(5, 3); |
| --- |

| int factorial(int n) {  if (n <= 1) return 1;  return n \* factorial(n - 1);  } |
| --- |