W05 Lab

Conditionals Part I

Sequential Program [Recap]

1. Prompt user for name
2. Read name
3. Prompt user for surname
4. Read surname
5. Prompt user for age
6. Read age

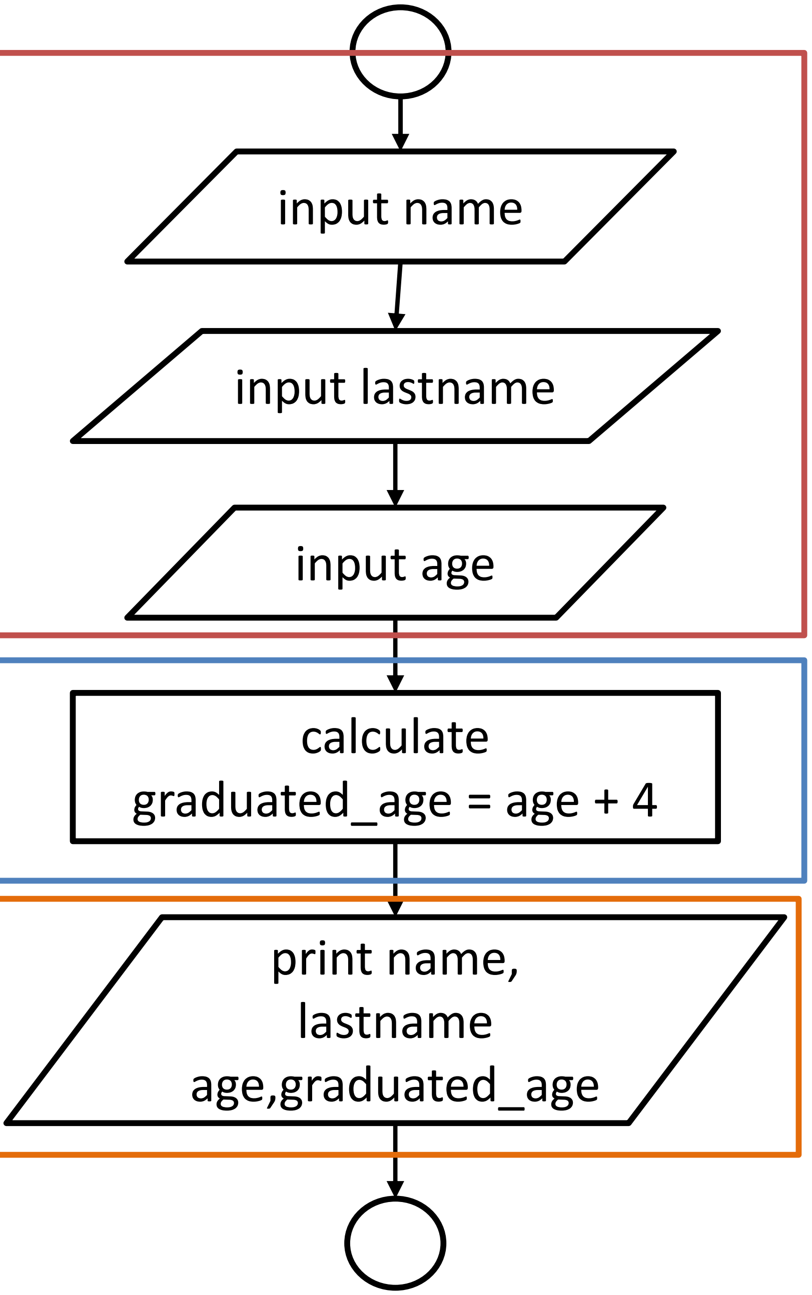
Input

Process

7. Calculate graduation age (age + 4)

8. Print name, surname, age, and graduation age

Output


input name
input lastname
input age
calculate graduated_age = age + 4
print name, lastname age,graduated_age

Statement and Expression

- a statement
  - a line of code or a command that performs an action or operation
- Expression
  - a combination of values, variables, operators, and function calls
  - Can be evaluated to produce a result.
  - It represents a computation or a calculation.

```
x = 5
y = 3
result = x + y * (x - 2)
```

Boolean Expression

- Evaluate to either True or False.
- Used in conditions, loops, and other control flow statements.
- Involve relational operators and logical operators to compare values or combine conditions

Relational Operators

- Equality (==): Checks if two values are equal.
- Inequality (!=): Checks if two values are not equal.
- Greater than (>): Checks if one value is greater than another.
- Less than (<): Checks if one value is less than another.
- Greater than or equal to (>=): Checks if one value is greater than or equal to another.
- Less than or equal to (<=): Checks if one value is less than or equal to another.

Relational Operators

- Conditions are usually have comparisons
- Let a = 10 and b = 20

Operator	Description	Example	result
`==`	`equal`	`a == b`	
`!=`	`not equal`	`a != b`	
`>`	`greater than`	`a > b`	
`<`	`less than`	`a < b`	
`>=`	`greater than or equal`	`a >= b`	
`<=`	`less than or equal`	`a <= b`	

Check Your Understanding

- What will print out?

```
x = 5
y = 10
z = 7

r1 = x < y
r2 = z >= y
r3 = x != z

print(r1,r2,r3)
```

Common evaluation: is_even()

- Check if the number is even number (e.g 2, 4, 6, 8)
- Create function is_even(num)
  - takes an integer parameter num
  - returns True if the number is even and False if it is odd.
  - Inside the function, use the modulo operator (%) to check if num % 2 equals 0.

Common evaluation: is_even()

- Complete this code on editor and run

```
____ is_even(num):

    result = ________
    return________




print(is_even(4))  #true
print(is_even(7))  #false
print(is_even(12)) #true
```

Voting Eligibility Check

- Write a function called `is_voting_age`
  - takes an integer parameter `age`
  - returns `true` if the person is of voting age (18 years or older)
  - returns `false` if they are not.
- Inside the function, use a boolean expression (age >= 18) to check if the age is greater than or equal to 18.
- If the condition is true, return True; otherwise, return False.
- Call the is_voting_age function with different test cases and print the returned value.

Common evaluation: is_voting_age()

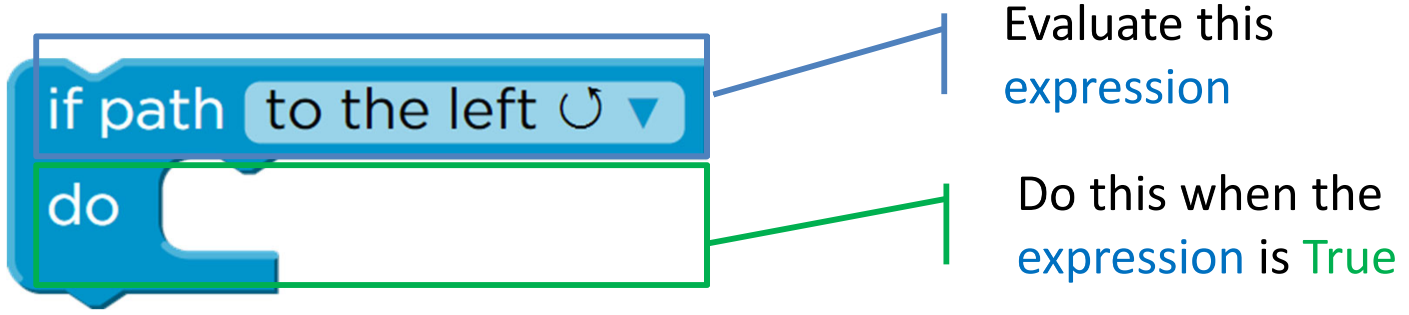
- Complete this code on editor and run

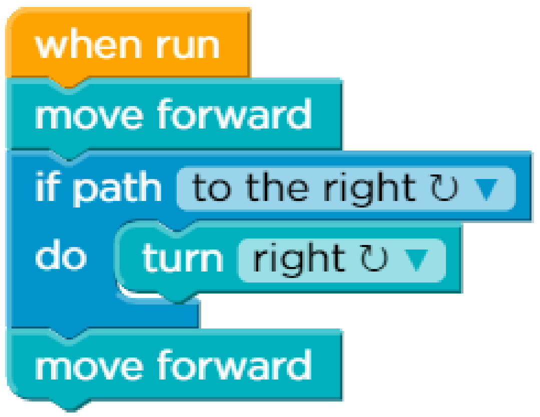
```
____ is_voting_age(age):

    result = ________
    return________



print(is_voting_age(16))    #false
print(is_voting_age(21))    #true
print(is_voting_age(18))    #true
```

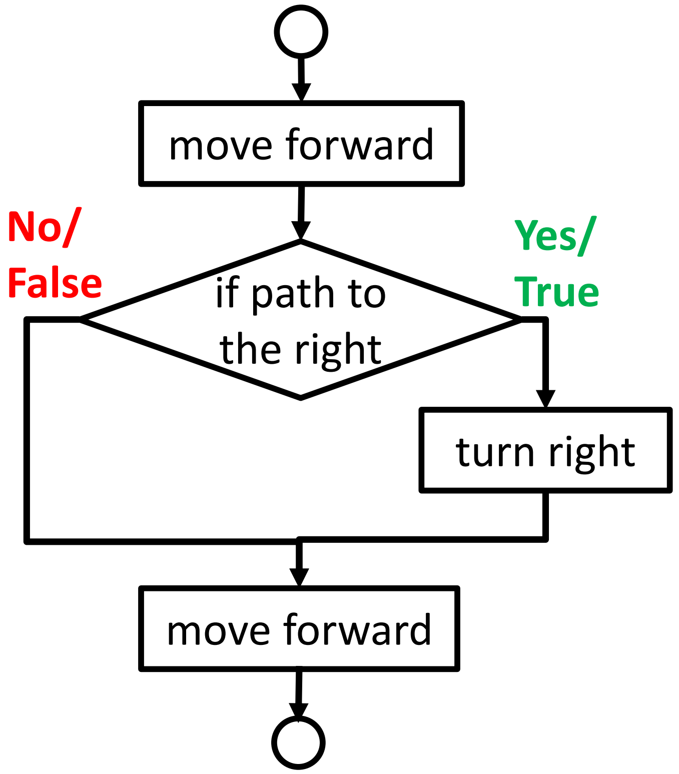
Condition Statement

- Expression or a statement that evaluates to either true or false
- Conditions are used to control the flow and make decisions.
- Condition is used to determine which block of code should be executed based on the evaluated result

Condition Statement

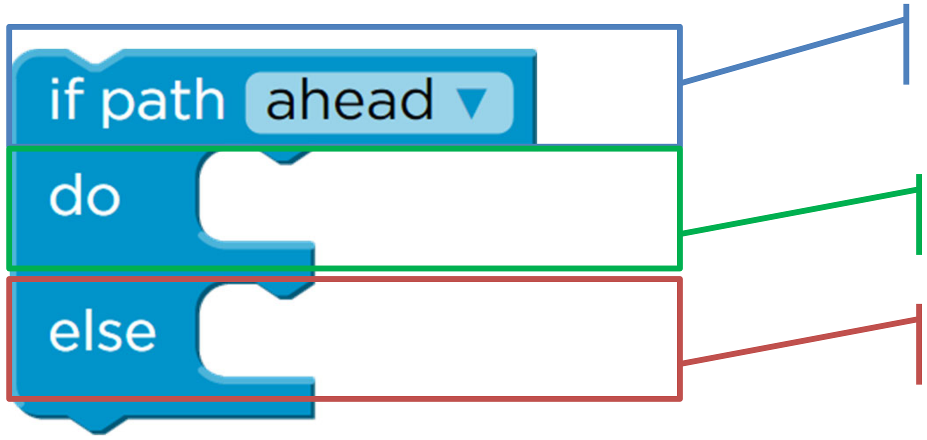






```
move_forward()
if  path_to_the_right:
        turn_right()
move_forward()
```

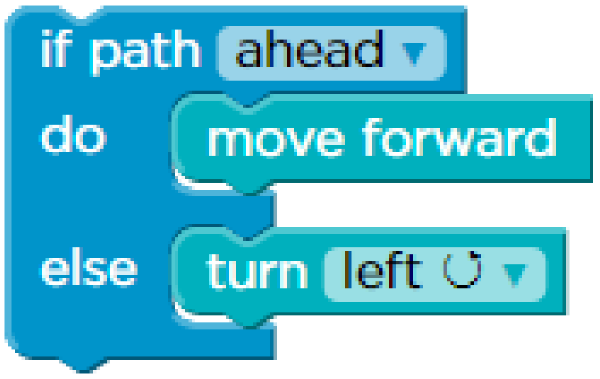
Condition Statement

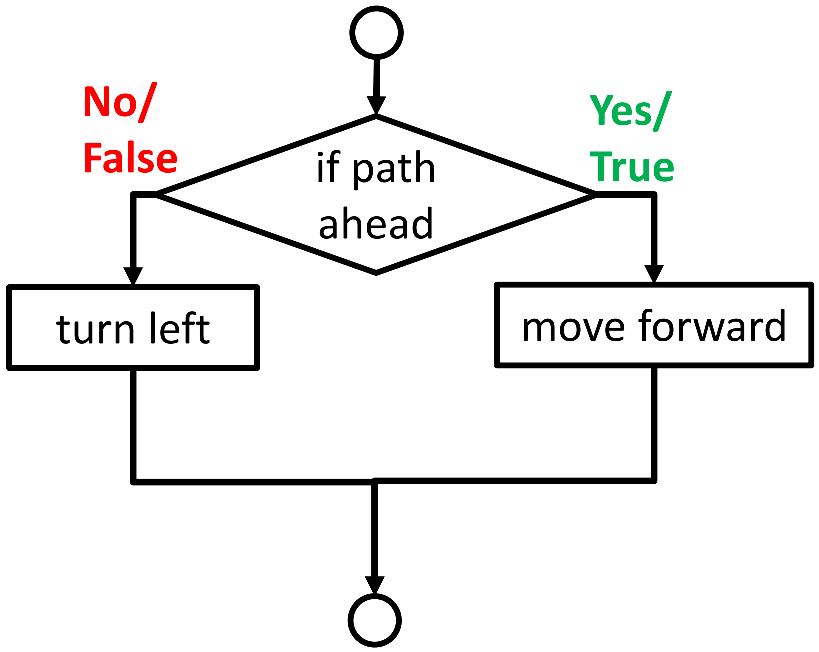


Evaluate this expression

Do this when the expression is True

Do this when the expression is False

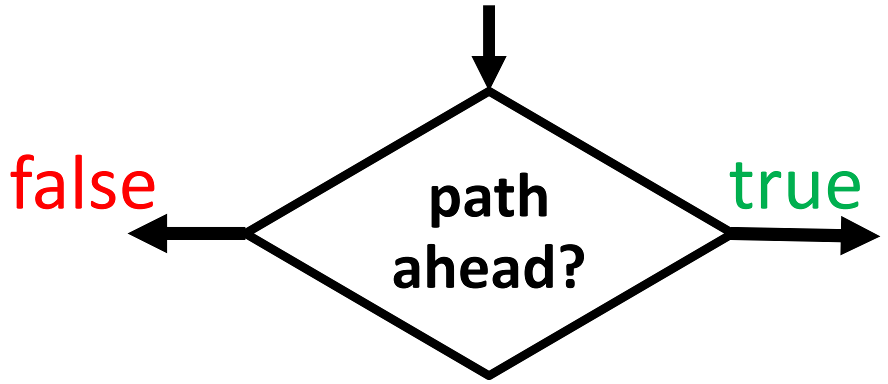


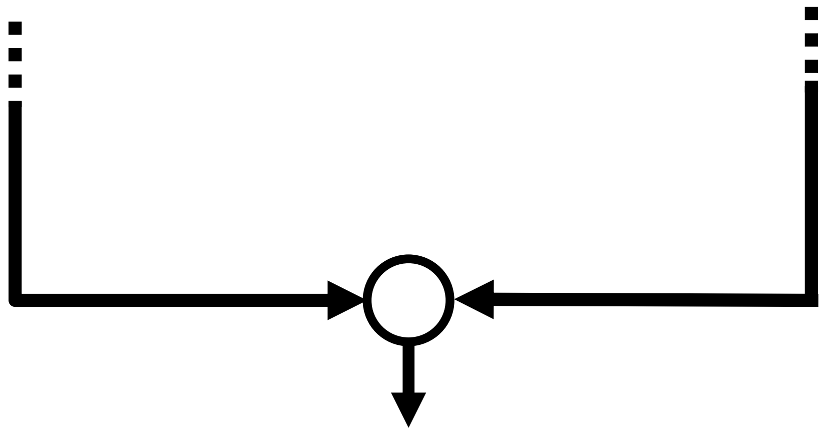


```
if  path_ahead:
        move_forward()
else:
     turn_left()
```

Decision Block

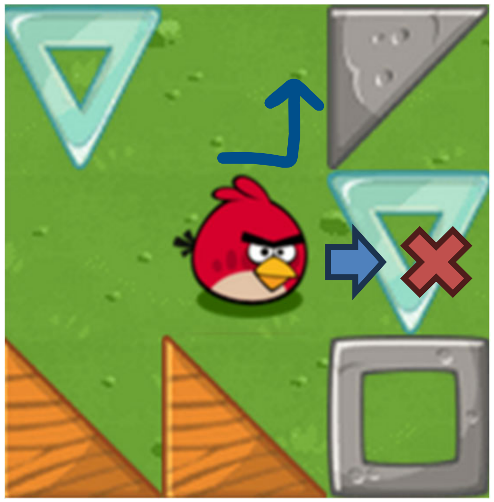
- Will allow the program to go on different path based on the condition provided.
  - If the condition is met (true), the program will continue on T- True path
  - If the condition is NOT met (false), the program will continue on F -False path instead.
- The paths may join back later on

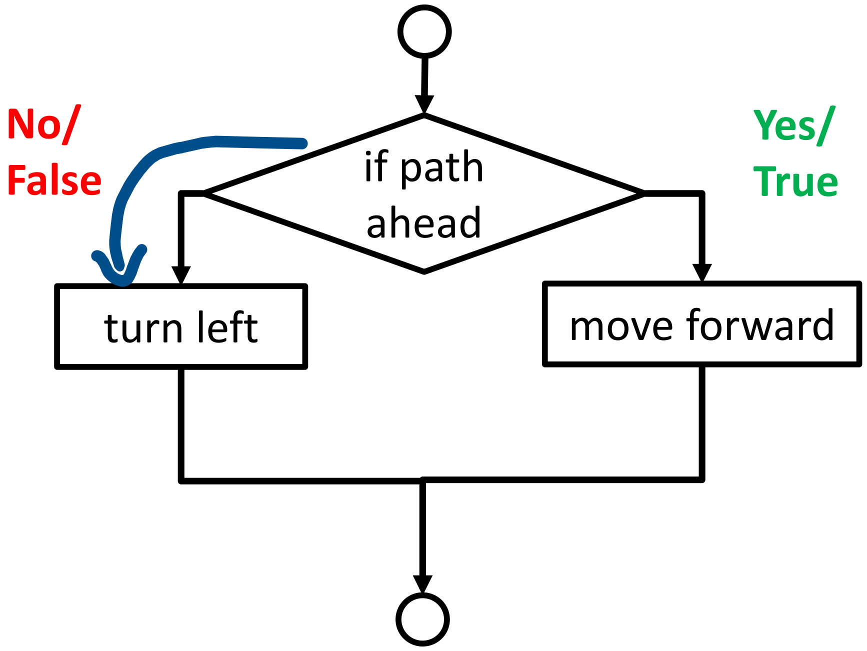




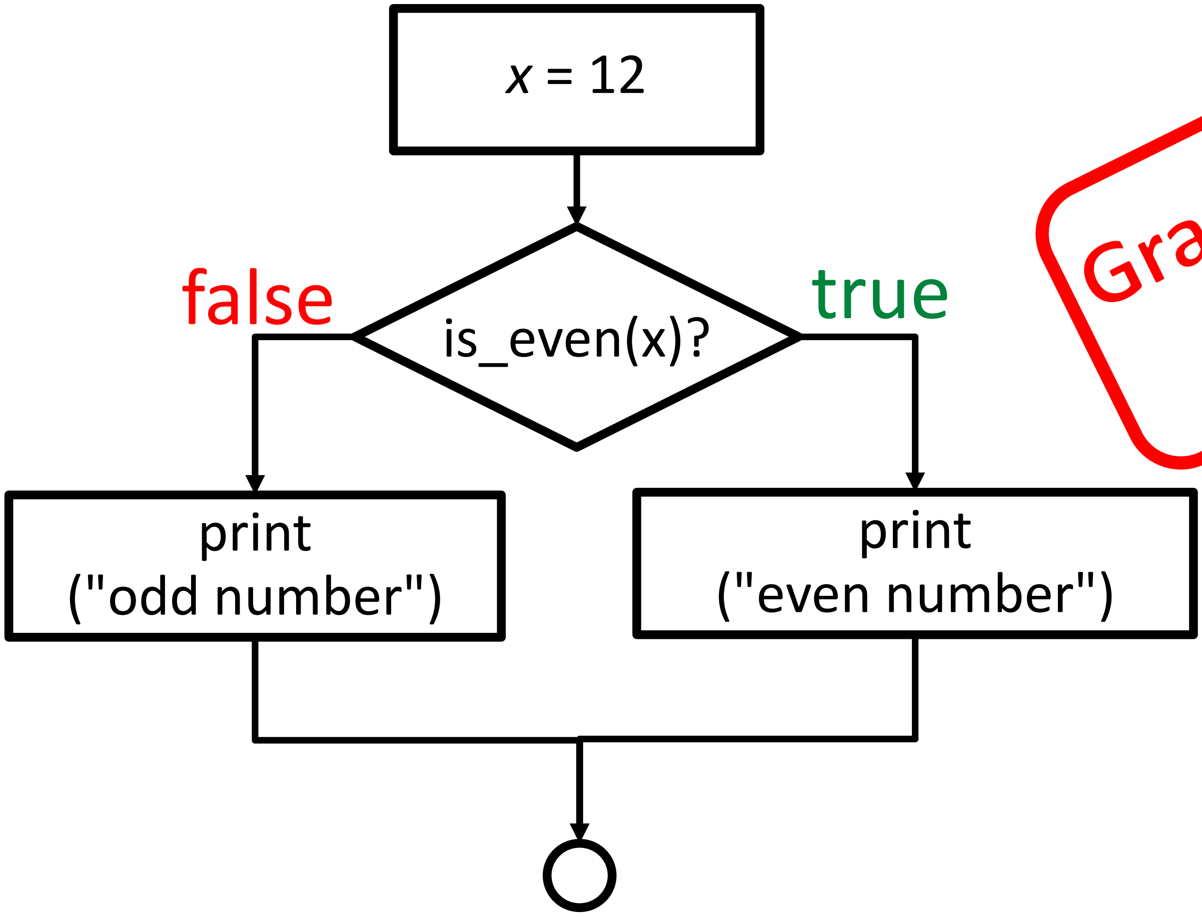
Selecting a Path

- Once a path is selected, the code/blocks on the other path will not be executed
- Unless looping is involved...





Conditional Flow control



Grader Alert!
w05_1

if statement

- if statement is a fundamental control structure in programming that allows the execution of certain code blocks based on a condition.

```
if condition:
    # code block to execute if the condition is true
```

```
x = 5
if x > 0:
    print("x is positive")
```

condition

don't forget :

run only if x>0 is true

4 spaces indentation

if-else statement

- The if-else statement allows a program to execute different code blocks based on a condition.
- It provides an alternative path of execution when if statement evaluates to false.

```
if condition:
    # code block to execute if the condition is true
else:
    # code block to execute if the condition is false
```

condition

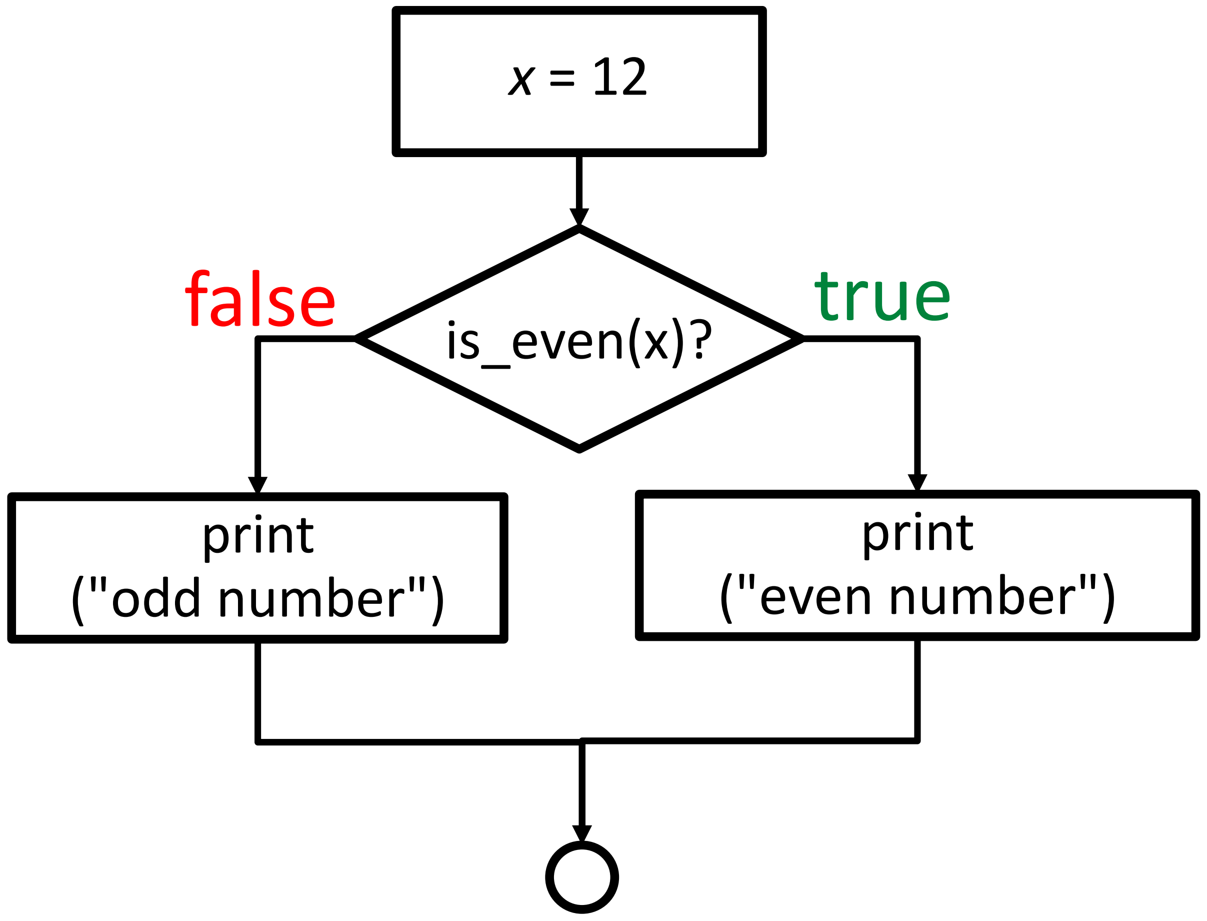
```
x = 5
if x > 0:
____print("x is positive")       → run only if x > 0
else:
____print("x is non-positive")   → run when x ≤ 0
                                          (condition is false)
```

4 spaces indent

don't forget :

Create Condition Flow with if-else

- Complete the code on the right hand side - reuse the is_even function for condition



```
def is_even(num):
…
…


x = 12
if ___________:
    print("odd number")
_____
    print("even number")
```

Create a function with Condition return

```
def function_name(parameters):
    # Code block
    return result
```

function definition

```
if condition:
    # code block to execute
    #if the condition is true
else:
    # code block to execute
    #if the condition is false
```

if - else condition

```
def function_name(parameters):

    if condition:
        result = "condition is true"
    else:
        result = "condition is false"

    return result
```

```
def function_name(parameters):

    if condition:
        return("condition is true")
    else:
        return("condition is false")
```

Create a function with Condition return

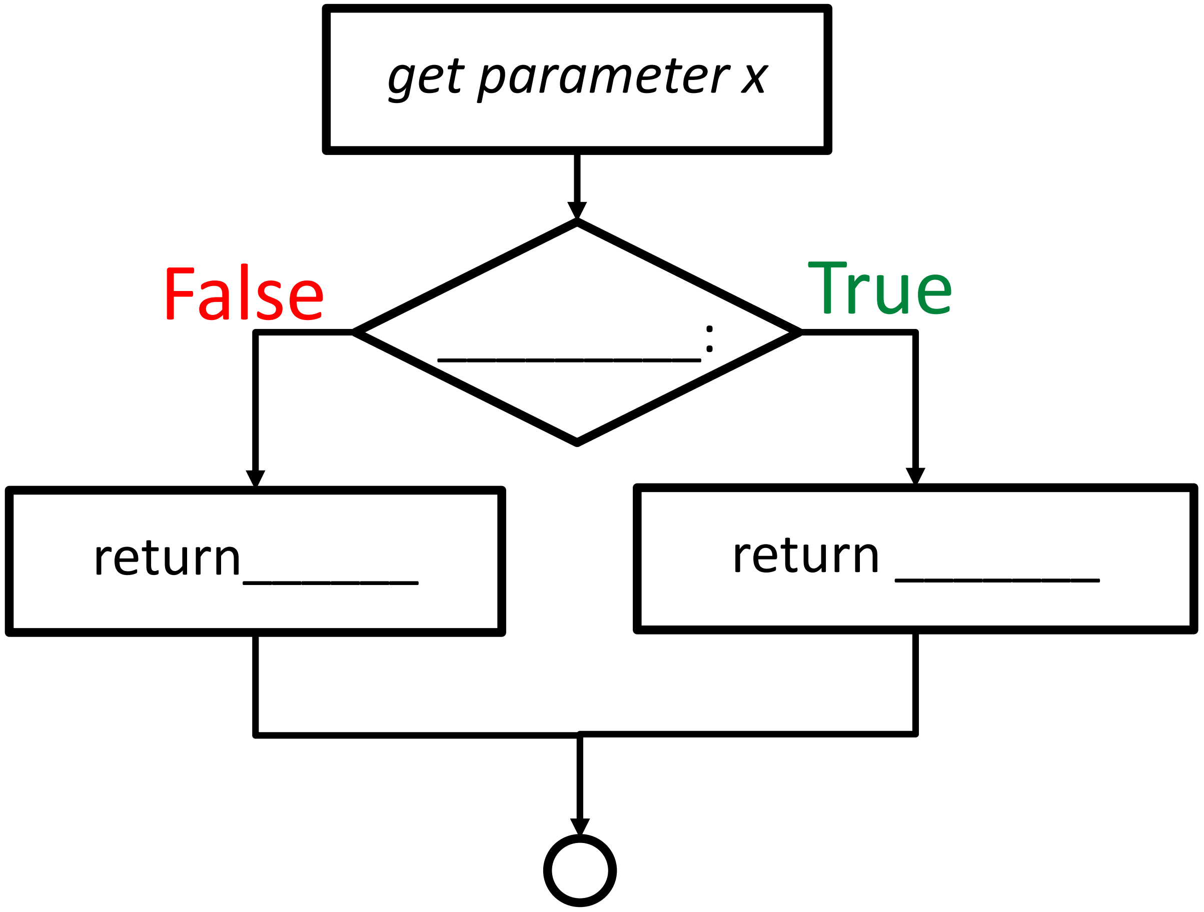
- The condition in a function can be based on various factors, such as the input parameters, intermediate calculations, or external variables.
- It allows the function to adapt its behavior and return different results based on different conditions.
- By using a function with condition return, you can encapsulate complex logic and decision-making within a reusable block of code.
- It promotes code reusability, modularity, and improves the overall readability and maintainability of your programs.

Create a function with Condition return

- Create a function named `is_positive` that takes 1 parameter: num.
- The function check if the parameter is positive or not
- return the result as Boolean value
    - True if num is positive number and return false if num Is negative number or zero

is_positive(num)

Grader Alert!
w05_2

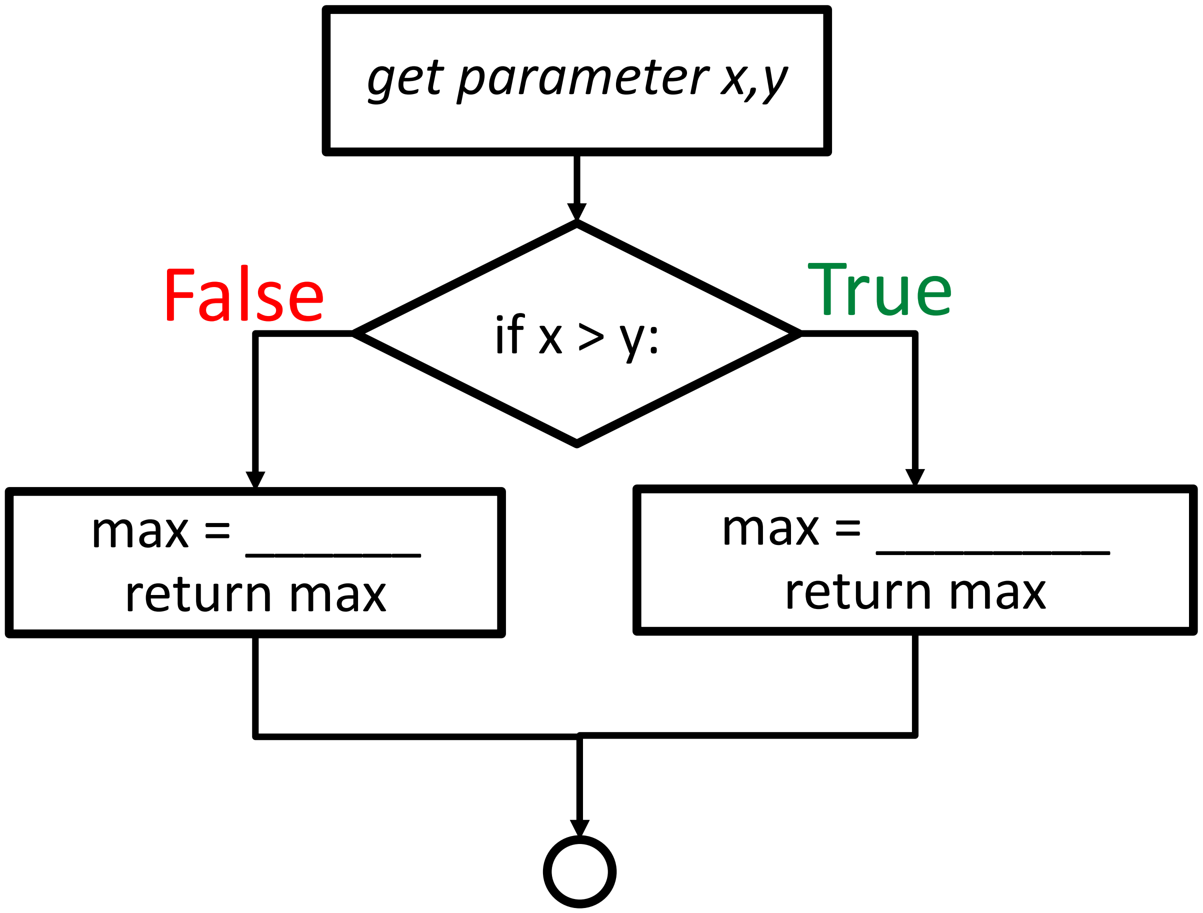


```
def is_positive(num):
    if ______________:
        return ___________
    else:
        return ___________
```

```
print(is_positive(34))
print(is_positive(-12))
print(is_positive(0))
print(is_positive(75))
```

Function with Condition

- Create a function that takes two numbers as input and returns the maximum of the two



Function with Condition

- Complete the function `find_max(num1,num2)`

```
def find_max(num1, num2):
if ________:
    return num1

______
    return _____


result = find_max(5, 10)
print("The maximum number is:", result)
```

Grader Alert!
w05_3

grader w05_3.py

Logical Operators

- AND (and): Checks if both conditions are true.
- OR (or): Checks if either condition is true.
- NOT (not): Negates the result of a condition.

Logical Operators

- Logical operators (such as and, or) can join multiple Boolean variables/values/comparisons together
- For example, assuming

```
a = true, b = false and c = true
```

Operator	Example	Result
`and`	`a and b`	
	`a and c`	
`or`	`a or b`	
`not`	`not(a and b)`	

Multiple condition with Logical Operators

- Create more complex conditions to control the flow of your program.
- It will evaluate multiple conditions simultaneously
- Make decisions based on their combined results.
- Consider operator precedence and use parentheses () to ensure the desired evaluation order.

Multiple condition with Logical Operators

- If both conditions are true (i.e., the number is between 0 and 50), the code block indented under the if statement is executed, printing the message "The number is between 0 and 50."
- If either one or both conditions are false (i.e., the number is outside the range of 0 to 50), the code block indented under the else statement is executed, printing the message "The number is outside the range of 0 to 50."

both condition must be true

```
def check_range(number):
    if number >= 0 and number <= 50:
        print("The number is between 0 and 50.")
    else:
        print("The number is outside the range of 0 to 50.")
```

indent for def

indent for if-else

```
print(check_range(15))
print(check_range(72))
print(check_range(33))
```

[redacted]

Check Your Understanding

- What will print out?

```
x = 5
y = 10
z = 7

r1 = x < y
r2 = z >= y
r3 = x != z

print(r1,r2,r3)
print(r1 and r2)
print(r2 or r3)
```

Multiple condition with Logical Operators

- Checking if a person is eligible for a discount.
  - a person can get a discount whenever has a membership or age over 60 years old
  - write a function get_discount(age, has_membership)
    - age is integer
    - has_membership is Boolean

get_discount()

- Complete the code below

```
def get_discount____________________________
    if _________________________________________:
        print("You are eligible for a discount.")
    else:
        print("You are not eligible for a discount.")
```