SFU SIMON FRASER UNIVERSITY
ENGAGING THE WORLD

CMPT 120: Introduction to Computing Science and Programming 1

Strings, and
Control Structures: if-elif-else

python™

Reminders

One-Stop Access To Course Information

- **Course website: One-stop access** to all course information.

  http://www2.cs.sfu.ca/CourseCentral/120/liaqata/WebSite/index.html

- Course Outline
- Exam Schedule
- Python Info
- CourSys/Canvas link
- Learning Outcomes
- Office Hours
- Textbook links
- and more…
- Grading Scheme
- Lab/Tutorial Info
- Assignments

- **Canvas**: Discussions forum - https://canvas.sfu.ca/courses/39187
- **CourSys**: Assignments submission, grades - www.coursys.sfu.ca

How to Learn in This Course?

A **Attend** Lectures & Labs

R **Read** / review Textbook/Slides/Notes

R **Reflect** and ask Questions

O **Organize** – your learning activities on weekly basis,

and finally…

W **Write** Code, Write Code, and Write Code.

Deliverables

1. Deliverables are due by the given date and time.
2. For the course, we are using IDLE to write and run our Python code.
3. You can use the CSIL lab computers outside your lab hours.
4. Plan ahead your assignments and other deliverables. Computer crash, network problems etc. are not acceptable excuses for delays in deliverables.
5. You may use online Python interpreters for running and testing your codes, such as:

   https://repl.it/languages/Python3

Labs

1. Each lab has an assigned TA.
2. Attend your assigned lab and show your work to your TA for the participation marks.
3. Class enrolments and lab swaps are closed now.

Course Topics

1. General introduction
2. Algorithms, flow charts and pseudocode
3. Procedural programming in Python
4. Data types and Control Structures
5. Fundamental algorithms
6. Binary encodings
7. Basics of computability and complexity
8. Basics of Recursion
9. Subject to time availability:
   - Basics of Data File management

Today's Topics

1. Strings

- String Special Operators
  - +, *, %
  - [], [:], in, not in
- String Formatting Symbols
  - %, s, d, m.n d
  - More symbols
- String Methods

2. Control Structures

- If statement
- Loop

1

Strings

String Special Operators

>>> word = "Welcome!"

- **[ ]** is called **slice** operator. We use to get a character from a string for given index. The index of first character is 0.
  - **Example:** **>>> print(word[0])**
    >>> _____
- **[:]** is called **range slice** operator. We use to get characters from a string for given index range.
  - **Example:** **>>> print(word[3 : 7])**
    >>> _____

String Operations 2

>>> **word = "Welcome!"**

- **in** is called **membership** operator. It returns **true** if a character exists in the given string.
  - **Example:** **>>> "e" in word** **>>> "k" in word**

- **not in** is called **membership** operator. It returns **true** if a character does not exist in the given string.
  - **Example:** **>>> "e" not in word** **>>> "k" not in word**

String Formatting Symbols

```
>>> course = "CMPT 120"
>>> print("Welcome to %s"  %course)
```

- **%s** format specifier is a placeholder for a string value.
- **%c** format specifier is a placeholder for a character.
- **%d** or **%i** format specifier is a placeholder for a signed decimal integer.
- **%u** format specifier is a placeholder for a unsigned decimal integer.
- **%f** format specifier is a placeholder for a floating point real number.
- **%o** format specifier is a placeholder for a octal integer.
- **%x** format specifier is a placeholder for a hexadecimal integer.
- **%e** format specifier is a placeholder for an exponent notation.

String Formatting Symbols – New Way {}

```
>>> course = "CMPT 120"
>>> print("Welcome to { }.".format(course))
Welcome to CMPT 120.

>>> course = "CMPT 120"
>>> mark = 87
print("Your mark in { } is { }.".format(course, mark))
```

String Methods

- `.upper()`: Convert a string to uppercase letters.
  - **Example:** **>>> "abc".upper()** ➔ _____
- `.strip()`: Removes leading and trailing spaces from a string.
  - **Example:** **>>> " abc ".strip()** ➔ _____
- `.isdigit()`: Returns true if string contains only digits and false otherwise.
  - **Example:** **>>> "abc".isdigit()** ➔ _____
- `.isnumeric()`: Returns true if a string contains only numeric characters and false otherwise.
  - **Example:** **>>> "abc ".isnumeric()** ➔ _____

String Methods 2

- `.lower()`: Convert a string to lowercase letters.
- `.lstrip()`: Removes leading spaces from a string.
- `.isspace()`: Returns true if string contains only whitespace characters.
- `.isalpha()`: Returns true if string has at least 1 character and all characters are alphabetic and false otherwise.
- **.capitalize()**: Capitalizes first letter of string.
- **len(string)**: Returns the length of the string.

2

Control Structures

Program Execution: Control Structures

Instructions in a program are executed in a sequential order from top to bottom, generally.

```
mid=input()
final = input()
sum = mid + final
print(sum)
```

- _____

Sometimes, we need to skip some instructions.

```
mid=input()
final = input()
sum = mid + final
if sum<50 :
    print("Fail")
else:
    print("Pass")
```

- _____: **Branching**

Sometimes, we need to repeat instructions.

```
sum = 0
n = 1
while (n <=100):
    sum=sum+n
    n=n+1
print(sum)
```

- _____ **Looping**

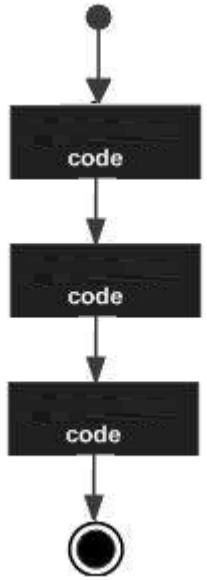
Control Structures

- **Control Structure**: It is a logical design which refers to the order in which statements in computer programs will be executed.

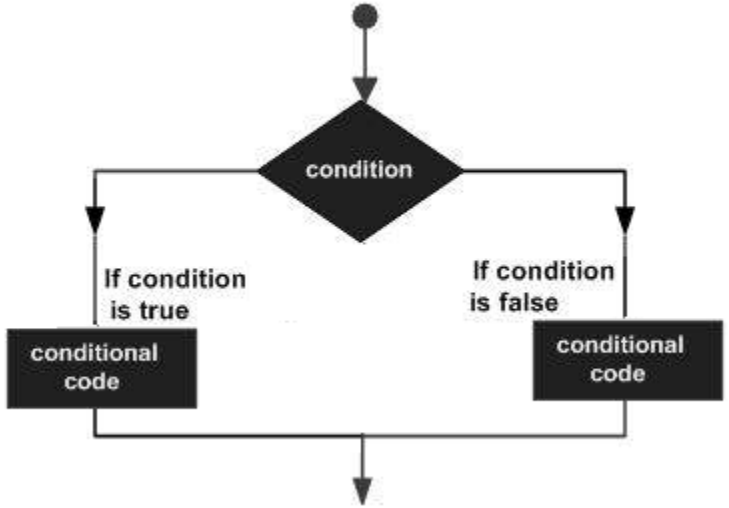
1. **Sequence Structure**: An order where a set of statements is executed sequentially.
2. **Decision Structure**: An order where a set of instructions is executed only if a condition exists.
   a. **Branching**
   b. **Looping**

Control Structures: Flowcharts

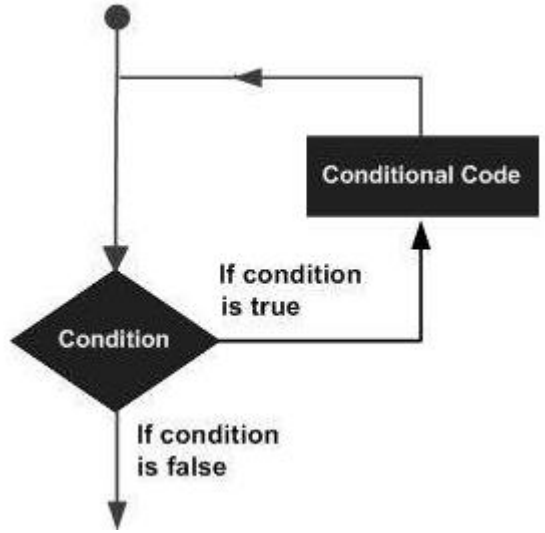
Sequential Structure



Decision Structure: Branching



Decision Structure: Looping



Decision Structures

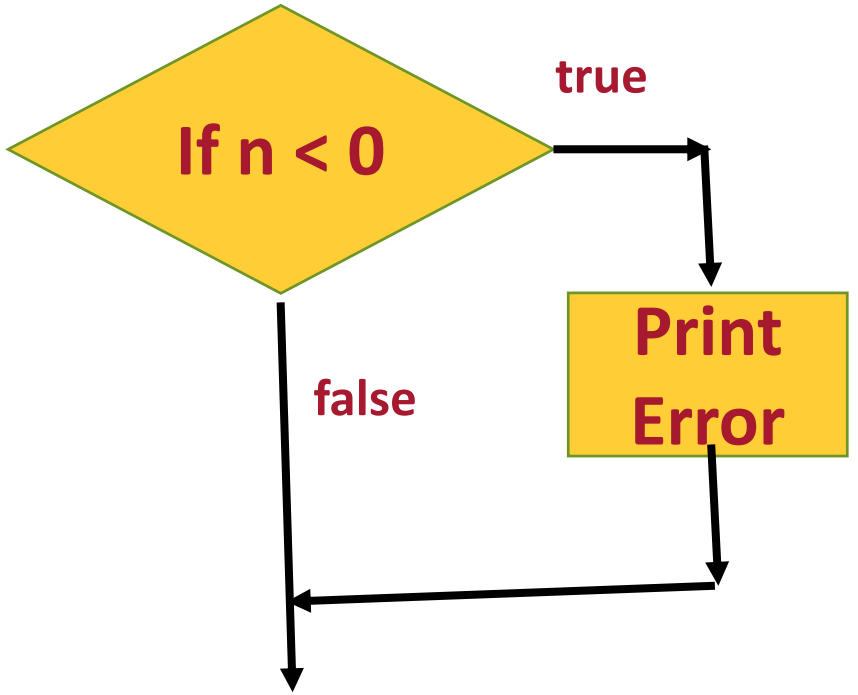
- **Branching**: It alters the flow of program execution by making a selection or choice.
  1. _____
  2. _____
  3. _____ **(A decision structure nested inside another decision structure)**
- **Looping**: It alters the flow of program execution by repetition of a particular block of statement(s).
  1. **for-loop**
  2. **while-loop**

3

The if Decision Structures

The if Statement: A Simple Decision Structure

- A simple if statement provides a **single** alternative decision structure.
  - It provides only one alternative path of execution.
  - If condition is not true, exit the structure.



The if Statement: Syntax

- Python syntax:

```
if condition:
    Statement
    Statement
```

- **First line** known as the _____.
- It includes the keyword **if** followed by _____.
- The condition can be **true** or **false**.
- When the **if statement** executes, the **condition is tested**, and if it is **true** the block statements are executed.
- Otherwise, block statements are skipped.

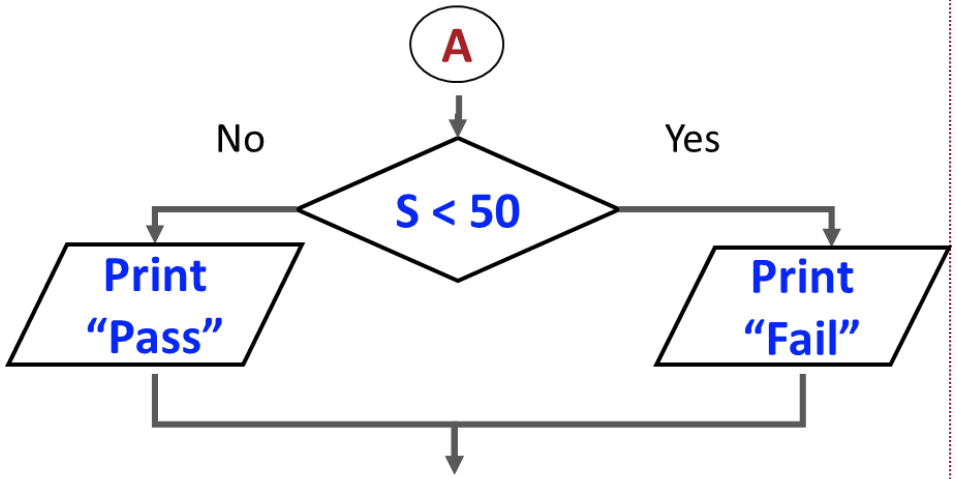
4

The if-else Decision Structures

The if-else Statement: Dual Alternative Decision Structure

- The **if-else** decision structure provides:
  - dual alternatives, or
  - two possible paths of execution.

1. One path is taken if the condition is true,
2. And, the other path is taken if the condition is false.



The if-else Statement: Syntax

- Python syntax:

```
if condition:
    Statement 1
    Statement 2
    Statement 3
else: condition:
    Statement 4
    Statement 5
    Statement 6
```

If the condition is true, this block of statements is executed.

If the condition is false, this block of statements is executed.

Then, control jumps here, to the statement following the `if-else` statement.

- **First line** known as the **if clause**.
- **Third** line known the **else clause**.
- The if clause and else clause must be **aligned**.
- **Statements** must be consistently **indented**.

5

The if-elif-else Decision Structures

The if-else Statement: Syntax

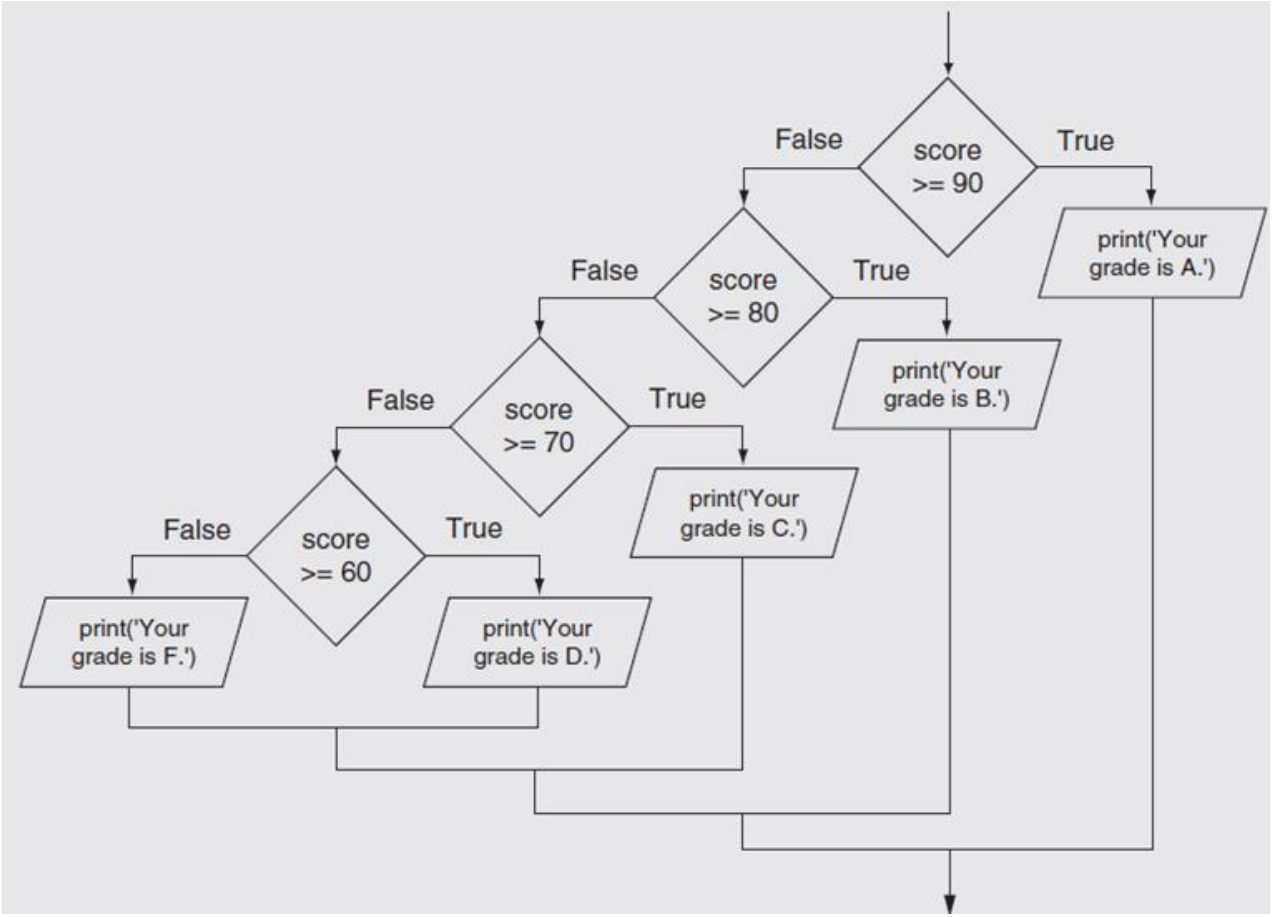
- The if-elif-else decision structure allows more than one condition to be tested.
- Python syntax:

```
if condition 1:
    Statement(s)
elif condition 2:
    Statement(s)
elif condition 3:
    Statement(s)
else:
    Statement(s)
```

Insert as many `elif` clauses as necessary.

- Use proper indentation in a nested decision structure.
- Indentation is important for Python interpreter, and enhance code readability.
- The **if, elif,** and **else** clauses must be **aligned**.
- **Statements** in each block must be consistently **indented**.
- The **if-elif-else** statement is never required, but it makes logic easier to follow.

The if-elif-else Statement: Grade Example



 Adapted.

The if-else Statement: Syntax

- The if-elif-else decision structure allows more than one condition to be tested.
- Python syntax:

```
if condition 1:
    Statement(s)
elif condition 2:
    Statement(s)
elif condition 3:
    Statement(s)
else:
    Statement(s)
```

Insert as many `elif` clauses as necessary.

- Use proper indentation in a nested decision structure.
- Indentation is important for Python interpreter, and enhance code readability.
- The **if, elif,** and **else** clauses must be **aligned**.
- **Statements** in each block must be consistently **indented**.
- The **if-elif-else** statement is never required, but it makes logic easier to follow.

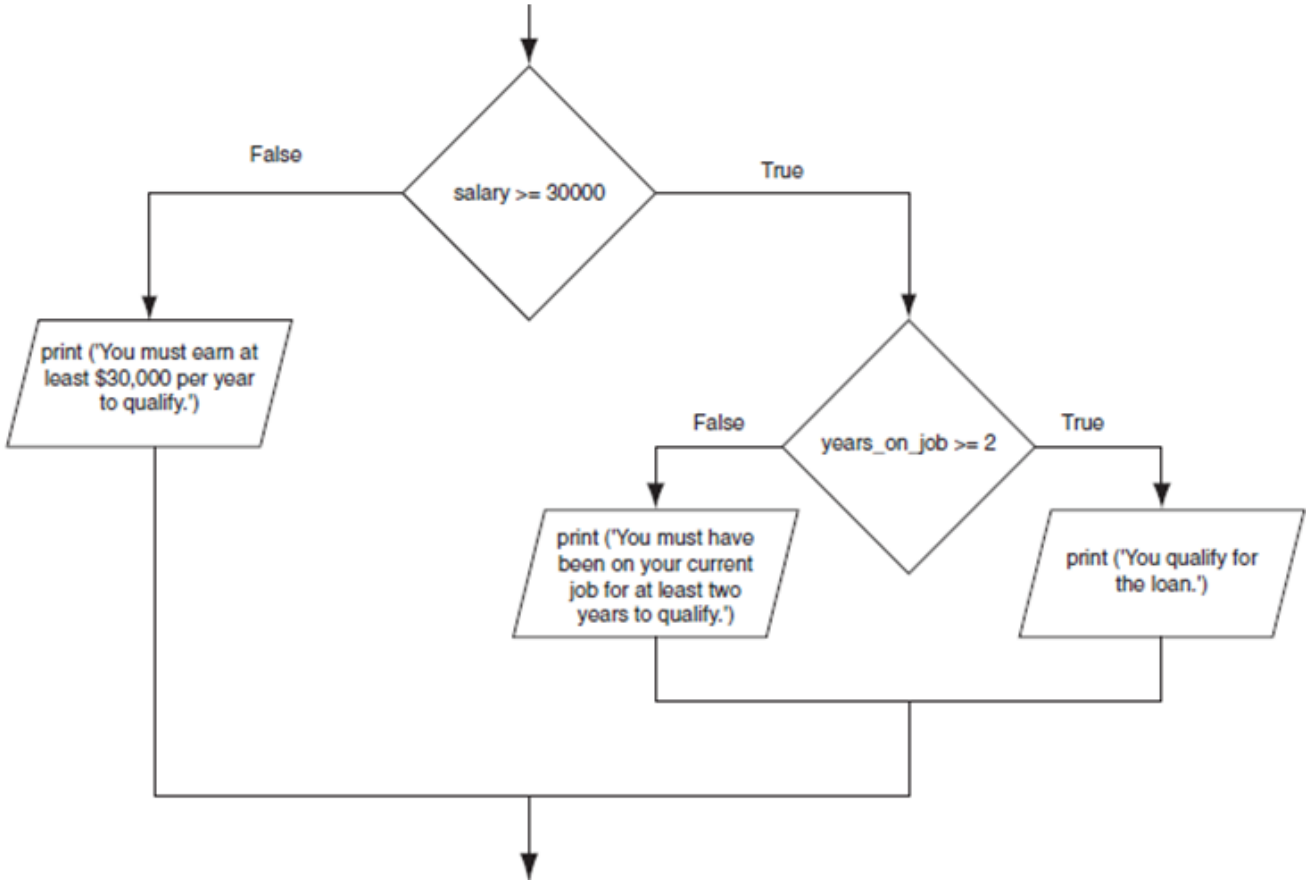
The if-elif-else Statement: Nested Decision Structure

- One condition or decision structure is nested inside another condition.

```
if condition 1:
    Statement(s)
    elif condition 2:
      Statement(s)
     else:
      Statement(s)
elif condition 3:
    Statement(s)
else:
    Statement(s)
```

- **Example:** Determine if someone qualifies for a loan, they must meet two conditions:
  - Must earn at least $30,000/year.
  - Must have been employed for at least two years.
- Check **first condition**, and if it is true, check **second condition**.

The if-elif-else Statement: Example Flowchart



 Adapted.

Example: What Lead Is Safe in Basketball?

- Bill James' Algorithm:

1. Take the number of points one team is ahead.
2. Subtract 3.
3. Add a half-point if the team that is ahead has the ball, and subtract a half-point if the other team has the ball. (**Numbers less than zero become zero.**)
4. Square that result.
5. If the result is greater than the number of seconds left in the game, the lead is safe.

Example: What Lead Is Safe in Basketball?

```
# 1. Take the number of points one team is ahead .
lead_str = _____________________________
lead_int= _____________________________
# 2. Subtract three .
lead_plus3 = _____________________
# 3. Add a half-point if the team that is ahead has the ball,
# and subtract a half-point if the other team has the ball .
has_ball = input("Does the lead team have the
                  ball (Yes or No):")
if has_ball == _____________:
  lead = _____________________
else:
  lead = _____________________
# (Numbers less than zero become zero )
if lead < 0:
   _________________
# 4. Square that .
lead_square  = _____________________
# 5. If the result is greater than the number of seconds left in the game,
# the lead is safe.
seconds = input("Enter the number of second remaining: ")
seconds_int = int(_____________)
if _____________________________:
   print("Lead is _____________")
else:
   print("Lead is _____________")
```

Class Participation: if-elif-else

- Write a Python program and post it on **Canvas by tonight 11:59pm**.
- Requirements:

1. Input a number from the user.
2. Use the if, elif, and else statements to check if the number is:
   i. positive, or
   ii. Negative, or
   iii. zero.
3. Display an appropriate message.

Questions?