Lecture 04

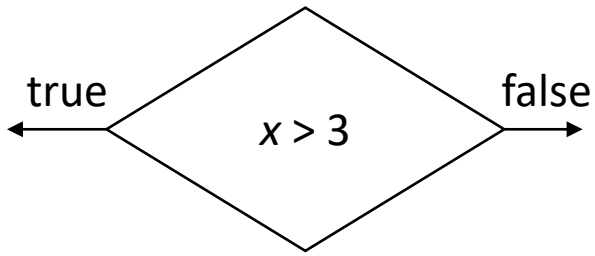
Conditional Structure

What is a conditional structure?

- A structure, defined in a programming language, that contains a conditional clause and its associated statements of the consequence(s).
  - A decision based on the given condition will be made to determine the consequence.
- In Python, it is referred to as the ***if*** statement.
  - Variations (*next few slides*)
- What about the ***switch..case*** statement?
  - As of Jan 2023, Python does not support *switch..case* statements
  - Programmers can simulate it using a dictionary

But, Python 3.10 introduces a new structure: match..case (too new to talk about it)

Basic *if..else* Structure



- The following illustrates a generic if structure

```
if (condition)
 codes to run when condition is true
else
 code to run when condition is false
```

- where,
  - *condition* must be Boolean expression

Basic Syntax of *if* statement

- Without ***else*** clause:

```
if (condition):
 statement(s) for true
```

- *condition* must be a Boolean expression. E.g. `(x < 5)`

```
if weight > 50:
   print("A $25 charge applies.")
```

- With ***else*** clause:

```
if (condition):
 statement(s) for true
else:
 statement(s) for false
```

```
temperature = float(input("What is the temperature? "))

if temperature > 70:
   print('Wear shorts.')
else:
   print('Wear long pants.')
```

The *elif* clause

- A Python way to combine else and if for saying "if the previous conditions were not true, then try this condition".
- You can have as many ***elif*** clauses in an ***if*** structure as needed.

```
n1 = float(input("Enter the 1st number: "))
n2 = float(input("Enter the 2nd number: "))

if (n1 > n2) :
    print(n1, "is larger.")
elif (n1 < n2):
    print(n2, "is larger.")
else:
    print("They are equal.")
```

elif $\neq$ *else if*

Python does not support “else if”

Nested *if* in Python

- The term “nested” means “one in another”.
- There may be a situation when you want to check for another condition after a condition resolves to true.
  - nested *if* could be a solution
- Nesting can happen within the *if* or *else* clauses. (next slide)

```
if (condition1):
 if (condition2):
  execution1
 else:
  execution2
else:
 executuin3
```

```
if (condition1):
 execution1
else:
 if (condition2):
  executuin2
 else:
  execution3
```

Sample codes

- Nesting can happen within the *if* or *else* clauses.

Score	Grade
90 ~	A
80 ~ 89.99	B
70 ~ 79.99	C
60 ~ 69.99	D
~ 60	F

```
gd = float(input("Enter a
score: "))

if (gd < 90):
  if (gd < 80):
    if (gd < 70):
       if (gd < 60):
         print("F")
       else:
         print("D")
    else:
      print("C")
  else:
    print("B")
else:
  print("A")
```

```
gd = float(input("Enter
a score: "))

if (gd >= 90):
  print("A")
else:
  if (gd >=80):
    print("B")
  else:
   if (gd >= 70):
     print("C")
   else:
     if (gd>=60):
       print("D")
     else:
       print("F")
```

Boolean expressions could be hybrid

- In Python, the Boolean expression of an if structure could be a hybrid expression.
  - “Hybrid” implies that two or more Boolean expressions are combined to form a single one

```
x = int(input("Enter an integer between 1 and 47: "))

if ((x < 2) or (x > 46)):
    print("Invalid number.")
else:
    print("Valid number.")
```

In-Class Exercise

- Write a program to calculate the total amount to pay after discount.

Cost	Discount
~ 25.99	0
26 ~ 44.99	5%
45 ~ 69.99	15%
70 ~	20%

```
tc = float(input("Enter the
total: "))

if (tc >= 70):
  tc = tc*(1-0.2)
else:
  if (tc >= 45):
    tc = tc*(1-0.15)
  else:
    if (tc >= 26):
      tc = tc*(1-0.05)
    else:
      tc = tc

print(tc)
```

An alternative of *switch..case* statement

- Python 3.x STILL does not support ***switch..case*** statement.
- Use a Python "dictionary" to simulate it.
  - A "dictionary" use a pair of *key* and *value* to define every element.
  - Let the *key* be the "case label", and let the *value* be the case statement.

```
///traditional
switch(gd)
{
 case 'A': str = "East more apples."; break;
 case 'B': str = "East more bananas."; break;
 case 'O': str = "East more oranges."; break;
 case 'AB': str = "East more avocados."; break;
}
```

```
####Python
switch = {'A': "Eat more apples.",
          'B': "Eat more bananas.",
          'O': "Eat more orange.",
          'AB': "Eat more avocados."}

gd = input("Enter your blood type: ")

print(switch[gd])
```

Section 2

switch..case

Sample code – *switch..case* simulation 1

- In Python, the “weekday()” function of the “datetime” module only returns an integer, not the name of week day, with 0 indicating Monday, 1 indicating Tuesday, an so on.

- Create a Python dictionary with the following elements to simulate a “*switch..case*” structure.
  - 0 is the “case label” of the case statement “Monday”.

```
switch = {
 0 : "Monday",
 1 : "Tuesday",
 2 : "Wednesday",
 3 : "Thursday",
 4 : "Friday",
 5 : "Saturday",
 6 : "Sunday" }
```

```
import datetime
now = datetime.datetime.now()

wd = now.weekday()

switch = { 0 : "Monday", 1 : "Tuesday",
         2 : "Wednesday", 3 : "Thursday",
         4 : "Friday", 5 : "Saturday",
         6 : "Sunday" }

print("Today is", switch[wd])
```

Sample code – *switch..case* simulation 2

- Write a Python program that will use a Python dictionary to simulate a *switch..case* structure to randomly display “tip of the day” based on the following table.

n	Tip
0	Failure is the path of lease persistence.
1	Now is the time to try something new.
2	Success is failure turned inside out.
3	You have yearning for perfection.
4	Those who care will make the effort.
5	Practice makes perfect.
6	Good news will come to you by mail.

```
import random

n = random.randint(0, 6)

switch = {
 0: "Failure is the path of lease persistence.",
 1: "Now is the time to try something new.",
 2: "Success is failure turned inside out.",
 3: "You have yearning for perfection.",
 4: "Those who care will make the effort.",
 5: "Practice makes perfect.",
 6: "Good news will come to you by mail." }

print("Tip of the day:", switch[n])
```

Sample code – *switch..case* simulation 3

- How to let the “case statement” perform a task?
  - In the previous example, the “case statement” simply return a text.
- Solution – let the value of the Python dictionary be an “function call” that can call a function.
  - A function in Python is a block of “Python code that performs the task” with a unique identifier.

```
def Add() : return (float(n1) + float(n2))
def Sub() : return (float(n1) - float(n2))
def Mul() : return (float(n1) * float(n2))
def Div() : return (float(n1) / float(n2))
def Mod() : return (float(n1) % float(n2))

switch = {
 "+": Add(),
 "-": Sub(),
 "*": Mul(),
 "/": Div(),
 "%": Mod(),
}
```

Combining *if* and *switch..case* in Python

- Scenario:
  - When the case label needs to be a range of values.

- E.g. A company offers discount of a product by quantity as shown below. The MSRP is $15.25. Write a Python program that ask the user to enter the quantity and return the selling price after discount.

Quantity	Discount Rate
1 ~ 100	0%
101 ~ 200	5%
201 ~ 300	10%
301 ~	15%

```
qty = int(input("Enter the quantity: "))

switch = { 'A': 0.15, 'B': 0.1, 'C': 0.05 }

if (qty >= 301):
    print(format(switch['A']*15.25*qty, ".2f"))
elif (qty <= 201):
    print(format(switch['B']*15.25*qty, ".2f"))
elif (qty <= 101):
    print(format(switch['C']*15.25*qty, ".2f"))
else:
    print(format(15.25*qty, ".2f"))
```

Section 3

match..case

The new "*match..case*" structure

- Python 3.10 introduced a new structure -- "match..case" which could be functionally similar to the traditional "switch..case".
- Interestingly, it does not require the "break" statement to end a case (next slide)
- Syntax:

```
match variable:
    case label1:
    case label2:
    .......
    _:
```

It uses an underscore (_) to denote the "default" case.

```
bt = input("Enter your blood type: ")

match bt:
  case "A": print("Eat apples.")
  case "B": print("Eat banana.")
  case "O": print("Eat orange.")
  case "AB": print("Eat apple and banana.")
  case _: print("No such blood type.")
```

Sample Codes (C++ and Java)

```
public class Sample
{
```

```
rgs[])

ood type: ");

t apples."); break;
t banana."); break;
t orange."); break;
at apple and banana."); break;
such blood type.");
```

```
#include <iostream>
using namespace std;

int main()
{
 int bt=0;
 cout << "Enter your blood type [0-A, 1-B, 2-O, 3-AB]: ";
 cin >> bt;

 switch (bt)
 {
  case 0: cout << "Eat apples." << endl; break;
  case 1: cout << "Eat banana." << endl; break;
  case 2: cout << "Eat orange." << endl; break;
  case 3: cout << "Eat apple and banana." << endl; break;
  default: cout << "No such blood type." << endl;
 }

 return 0;
}
```