Module 3: Branching

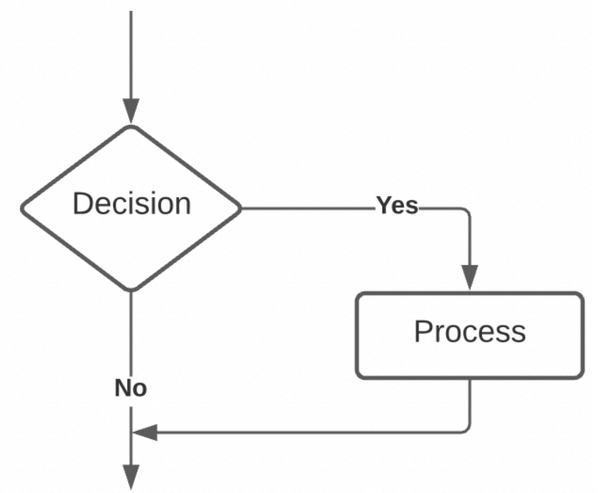
- We learned what programming languages are, and why we use Python.
- We learned how to use print() and input() to build interactive programs for simple input/output.
- We also learned variables and operations.
- We are going to learn more flow control methods.
  - Branching
  - Repetition

Flow Control

- Flow control is the logical design that controls order in which set of statements execute
- So far, the programs we wrote, are executing line by line - or Sequence Structure.
- We need more:
  - Branching - Decision Structure
  - Repetition - Looping Structure

Branching

- We can have specific action(s) performed only if a condition satisfied.
  - Also known as Decision Structure, or Selection Structure.



The `if` Statement

- We use `if` to start a branching:

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

- The first line known as the `if` clause:
  - Includes the keyword `if`
  - Specifies the *`condition`* with a `:`
  - List statement(s) with one indention.
  - Only when the condition is tested to be True, the statement(s) will be executed; otherwise, the block will be skipped.

Example: Absolute Value

- We can use `if` to convert any number x to absolute value.

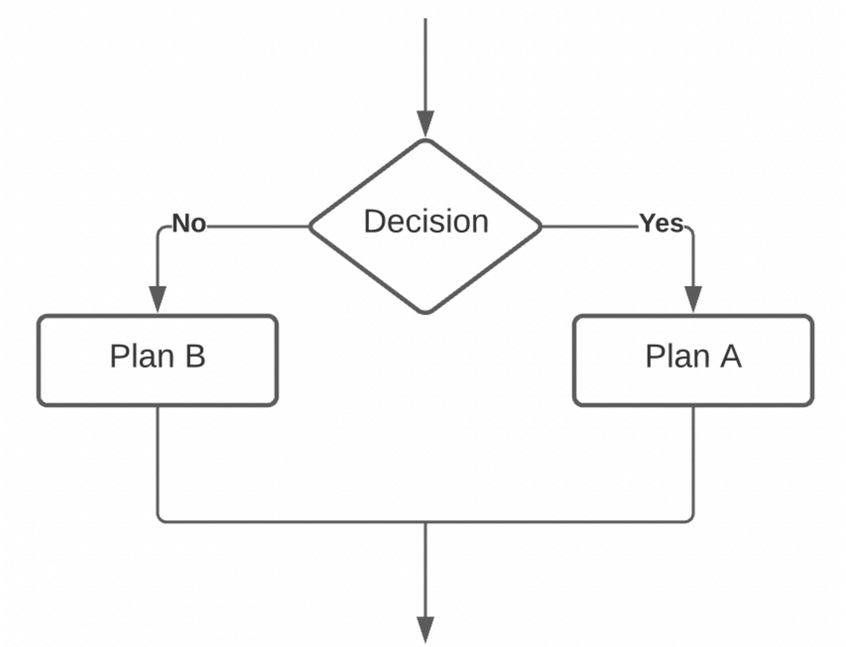
```
# computer the absolute value

x = int(input('Please enter a number:'))

if x < 0:
  x = -x

print('abs(x) is', x)
```

Let’s Play in Colab

- Download the M3Lab1.ipynb file.
- Upload it to your Colab.
- Finish the tasks.
- Use the discussion board to ask for help.

Branching

- We can have alternative action(s) performed based on a condition:



The `if-else` Statement

- We can also use if-else to start a branching:

```
if condition:
  statements
else:
  statements
```

- `if` clause and `else` clause must be aligned:
  - `if` clause specifies the *`condition`* with a `:`
  - `else` clause with a `:`
  - List statement(s) with consistent indention
  - If the condition is tested to be True, the statement(s) in `if` clause will be executed; otherwise, the statement(s) in `else` clause will be executed.

Example: Even or Odd

- We can use `if-else` to check an integer x is even or odd.

```
x = int(input('Please enter an integer:'))
if x % 2 == 0:
  print(x,'is even.')
else:
  print(x, 'is odd')
```

Example: Prime or Not

- We can use `if-else` to check an integer x within 10 is prime or not

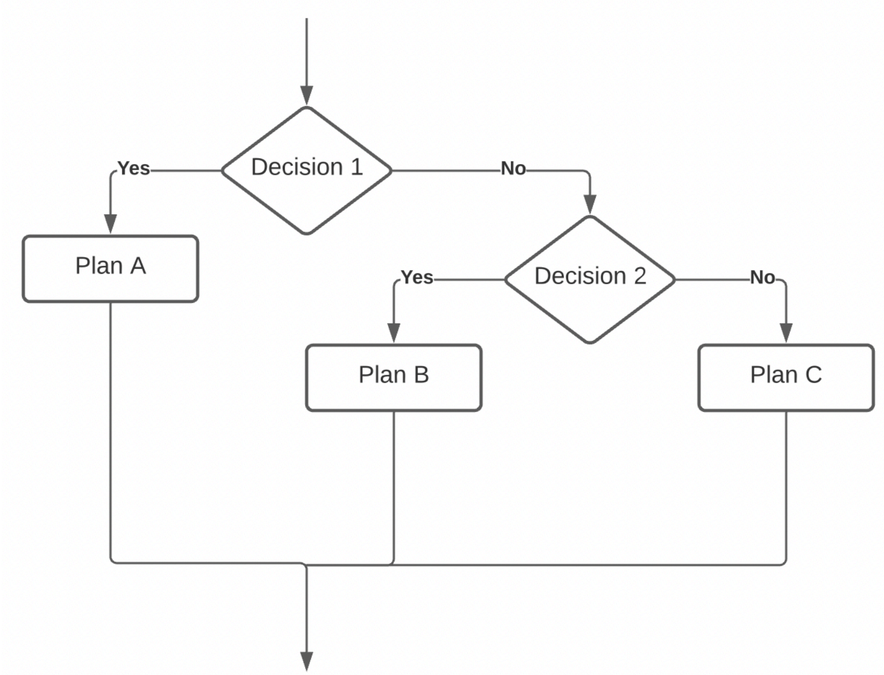
```
x = int(input('Please enter an integer within [1, 10]:'))
if x == 1 or x == 2 or x == 3 or x == 5 or x == 7:
  print(x, 'is prime.')
else:
  print(x, 'is not prime')
```

Let’s Play in Colab

- Download the M3Lab2.ipynb file.
- Upload it to your Colab.
- Finish the tasks.
- Use the discussion board to ask for help.

Branching

- We can have a specific action(s) performed based on a nested condition:



The `if-elif-else` Statement

- We can also use if-elif-else to start a branching:

```
if condition1:
  statements
elif condition2:
  statements
else:
  statements
```

- `if` clause, `elif` clause, and `else` clause must be aligned:
  - `if` clause and `elif` clause specify the *`conditions`* with a `:`
  - `else` clause with a `:`
  - List statement(s) with consistent indention
  - If the condition is tested to be True, the statement(s) in `if/elif` clause will be executed; otherwise, the statement(s) in `else` clause will be executed.

Example: Hiking?

- We can use `if-elif-else` to check whether it is a good day for hiking:

```
x = float(input('Please enter the temporature in Fahrenheit:'))

if x < 60:
  print('It is too cold! Be careful!')
elif x > 80:
  print('It is too hot! Be careful!')
else:
  print('Perfect!')
```

Example: Letter Grade?

- We can use `if-elif-else` to convert a point-grade into a letter grade:

```
x = float(input('Please enter the point-grade in [0, 100]:'))

if x >= 90:
  grade = 'A'
elif x >= 80:
  grade = 'B'
elif x >= 70:
  grade = 'C'
elif x >= 60:
  grade = 'D'
else:
  grade = 'F'

print('The letter-grade is:', grade)
```

Let’s Play in Colab

- Download the M3Lab3.ipynb file.
- Upload it to your Colab.
- Finish the tasks.
- Use the discussion board to ask for help.

Let’s Play in Colab

- Download the M3Assignment.ipynb file.
- Upload it to your Colab.
- Finish the tasks.
- Submit your assignment.
- Use the discussion board to ask for help.

Congratulations!

- You finished Module 3.
- Programming in Python is still not that hard, right?
- We are going to learn repetition in next module and get even more power unlocked!
- See you soon!