SCHOOL OF
INTERNATIONAL & INTERDISCIPLINARY
ENGINEERING PROGRAMS
King Mongkut's Institute of Technology Ladkrabang

Python Crash Course

Dr. Sirasit Lochanachit

01526102 Data Structures and Algorithms

Outline

- Python Overview
- Objects in Python
- Operators and Precedence

Python Overview

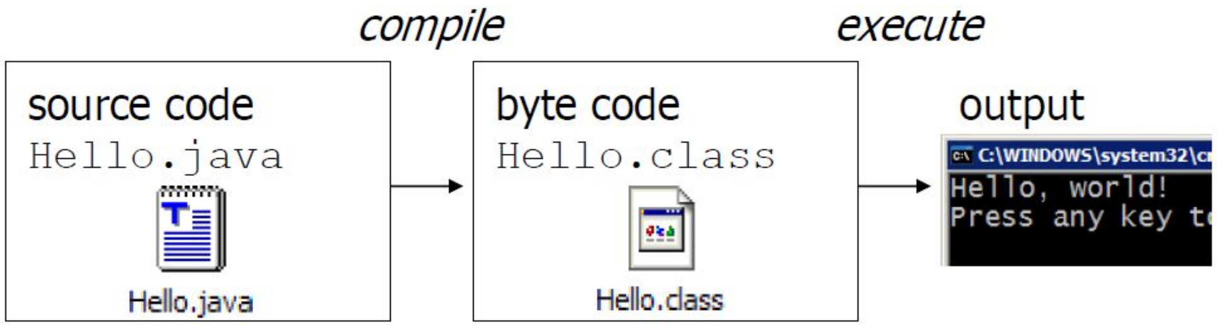
Python is formally an interpreted language.

Commands are executed through a piece of software known as the Python interpreter.

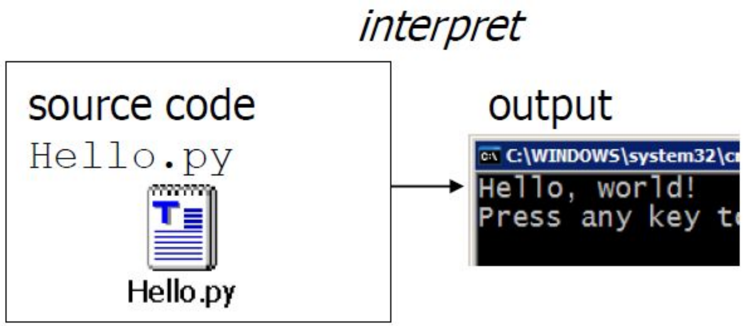
- The interpreter receives a command -> evaluate -> reports the result

Programmer typically defines a series of commands in advance and saves those commands in a plain text file known as source code or a script.

For Python, source code is conventionally stored in a file named with the .py suffix (e.g., demo.py).



```
Hello, world!
Press any key t
```



```
Hello, world!
Press any key t
```

Objects in Python

Python is an object-oriented language and classes form the basis for all data types.

Python's built-in classes:

- **int** class for integers,
- **float** class for floating-point values,
- **str** class for character strings.

Assignment Statement

In Python, a variable can store any type of value/data without the need to declare a data type in advance.

- name = “Jane”
- age = 25

temperature = 35.6

(identifier/variable) (object)

Identifiers / Variables

Variables in Python are case-sensitive.

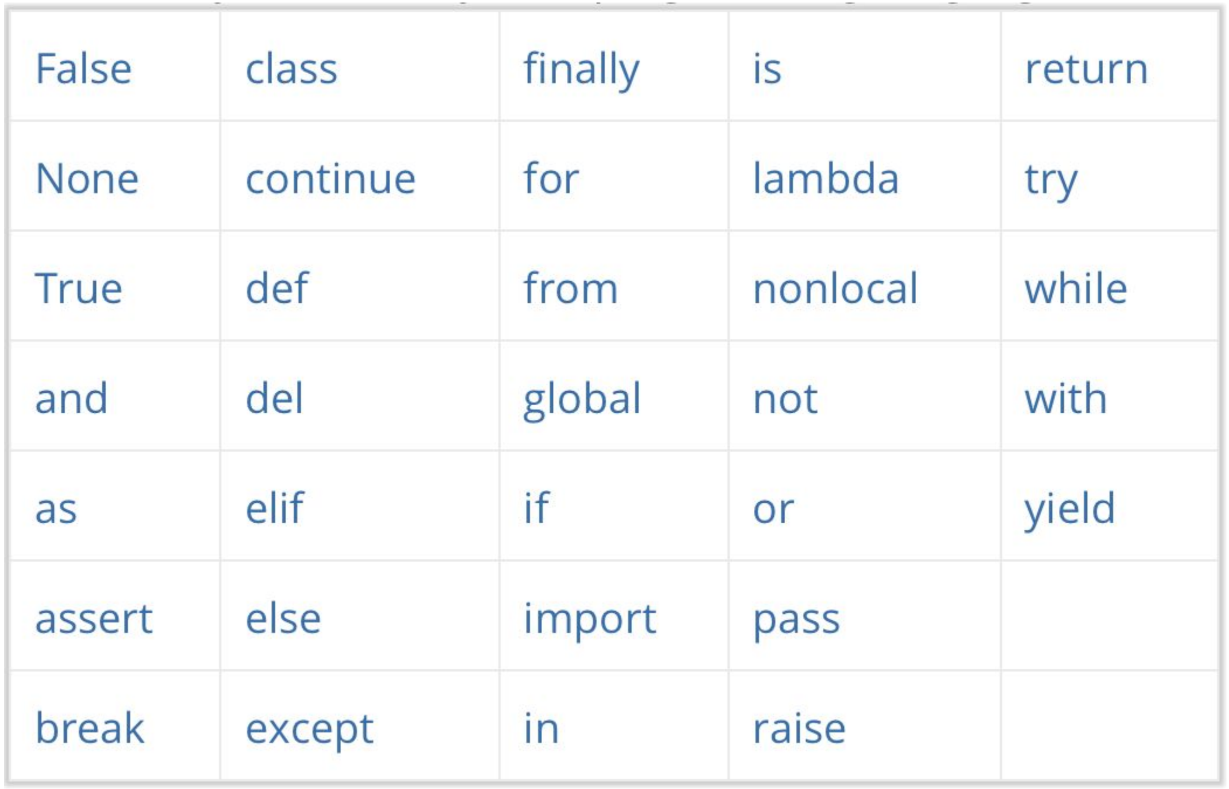
Variables can be composed of almost any combination of letters, numerals, and underscore characters.

- A variable cannot begin with a numeral

There are 33 specially reserved words that cannot be used as variables.

SCHOOL OF INTERNATIONAL & INTERDISCIPLINARY ENGINEERING PROGRAMS
King Mongkut's Institute of Technology Ladkrabang

Reserved Words

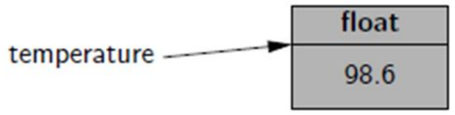


False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Dynamically Typed

temperature = 35.6

(identifier/variable) (object)

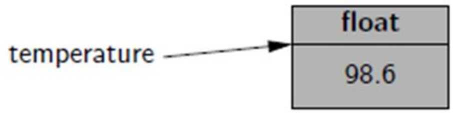


- Each identifier is implicitly associated with the memory address of the object to which it refers.
- No advance declaration associating a variable with a particular data type.
- A variable can be associated with any type of object, and it can later be reassigned to another object of the same (or different) type.

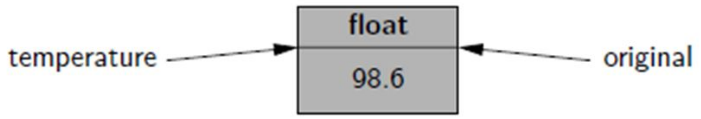
Dynamically Typed

temperature = 35.6

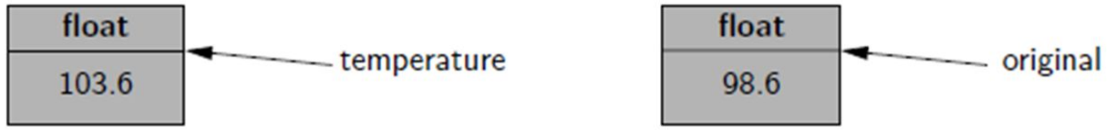
(identifier/variable)   (object)



original = temperature



temperature = temperature + 5



- A programmer can establish an alias by assigning a second identifier to an existing object.

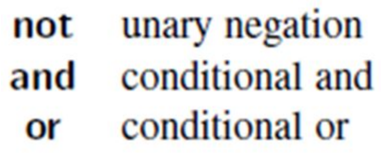
Built-in Python Data Types

Text Type:	str
Numeric Types:	int, float, complex
Sequence Types:	list, tuple, range
Mapping Type:	dict
Set Types:	set, frozenset
Boolean Type:	bool
Binary Types:	bytes, bytearray, memoryview
None Type:	NoneType

Ref: https://www.w3schools.com/python/python_datatypes.asp

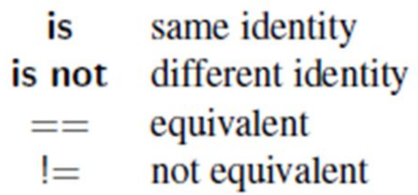
Operators and Precedence

- Logical Operators



not	unary negation
and	conditional and
or	conditional or

- Equality Operators



is	same identity
is not	different identity
==	equivalent
!=	not equivalent

Operators and Precedence

- Comparison Operators

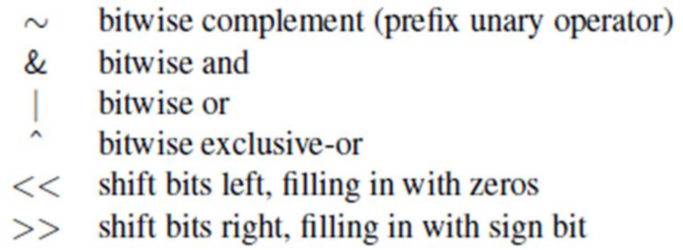
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to

- Arithmetic Operators

+	addition
−	subtraction
*	multiplication
/	true division
//	integer division
%	the modulo operator

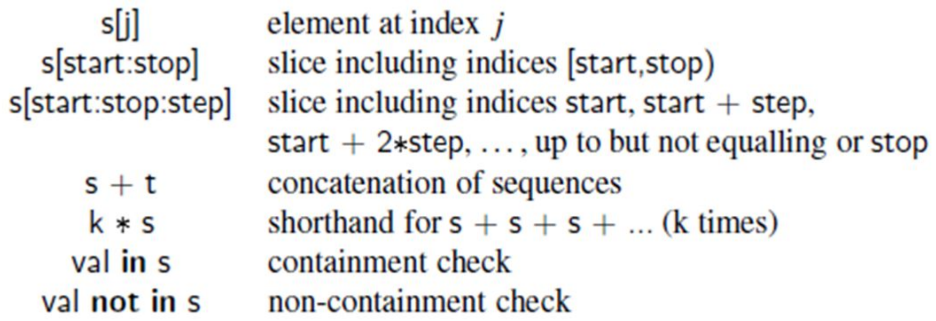
Operators and Precedence

- Bitwise Operators



~	bitwise complement (prefix unary operator)
&	bitwise and
|	bitwise or
^	bitwise exclusive-or
<<	shift bits left, filling in with zeros
>>	shift bits right, filling in with sign bit

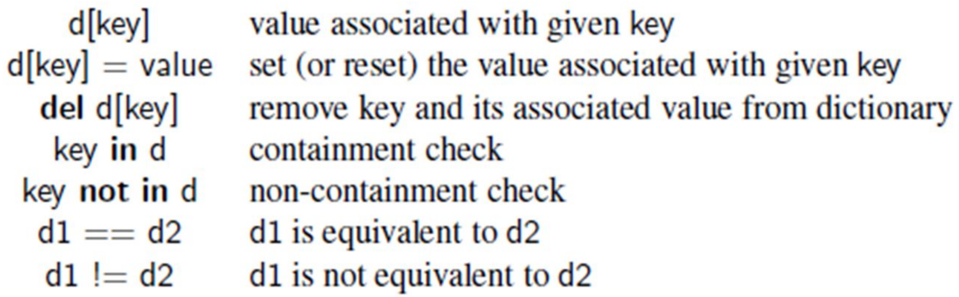
- Sequence Operators



s[j]	element at index *j*
s[start:stop]	slice including indices [start,stop)
s[start:stop:step]	slice including indices start, start + step, start + 2∗step, . . . , up to but not equalling or stop
s + t	concatenation of sequences
k ∗ s	shorthand for s + s + s + ... (k times)
val **in** s	containment check
val **not in** s	non-containment check

Operators and Precedence

- Dictionary Operators

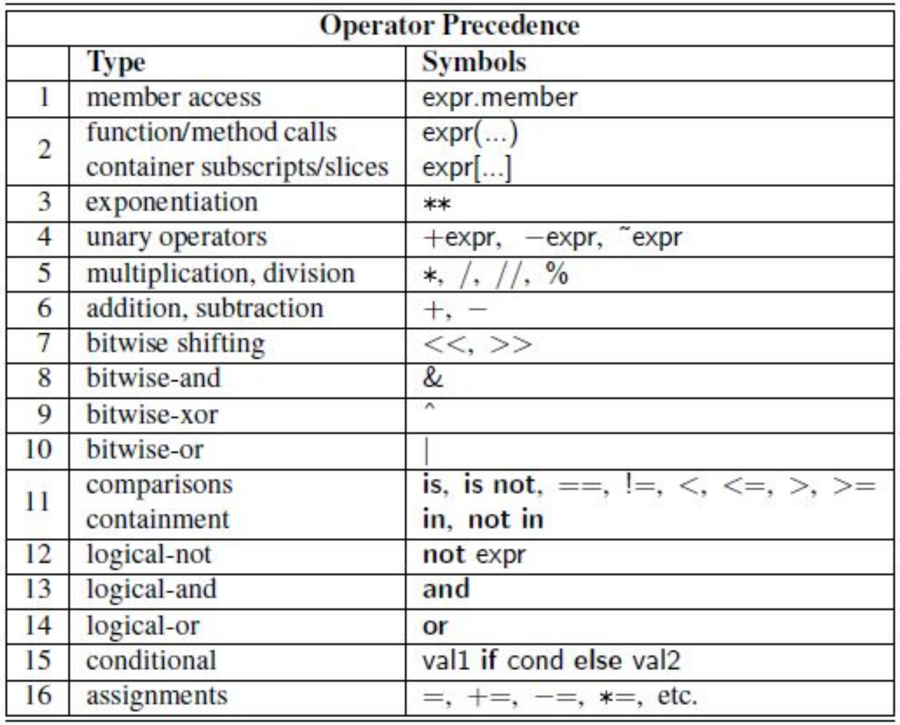


d[key]	value associated with given key
d[key] = value	set (or reset) the value associated with given key
del d[key]	remove key and its associated value from dictionary
key **in** d	containment check
key **not in** d	non-containment check
d1 == d2	d1 is equivalent to d2
d1 != d2	d1 is not equivalent to d2

Operators and Precedence

- Operator Precedence

Ordered from highest to lowest



Operator Precedence		
	Type	**Symbols**
1	member access	expr.member
2	function/method calls container subscripts/slices	expr(...) expr[...]
3	exponentiation	∗∗
4	unary operators	+expr, −expr, ˜expr
5	multiplication, division	∗, /, //, %
6	addition, subtraction	+, −
7	bitwise shifting	<<, >>
8	bitwise-and	&
9	bitwise-xor	^
10	bitwise-or	|
11	comparisons containment	is, is not, ==, !=, <, <=, >, >= in, not in
12	logical-not	not expr
13	logical-and	and
14	logical-or	or
15	conditional	val1 if cond else val2
16	assignments	=, +=, −=, ∗=, etc.

Exercise 1

Write a program to calculate an age based on a birth year (AD)

- Example input: 2001
- Expected output:
  - Your age is 22

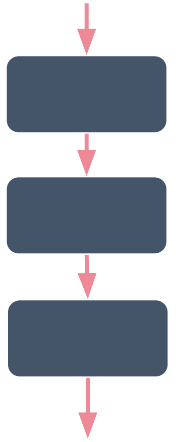
Exercise 2

Write a program to calculate an area of a triangle by accepting a height and a width (base) of a triangle.

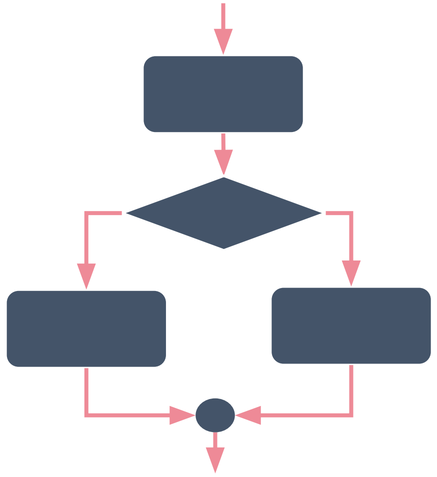
- Example input: triangle(10, 12)
- Expected output:
    - A triangular area is 60

SCHOOL OF INTERNATIONAL & INTERDISCIPLINARY ENGINEERING PROGRAMS
King Mongkut's Institute of Technology Ladkrabang

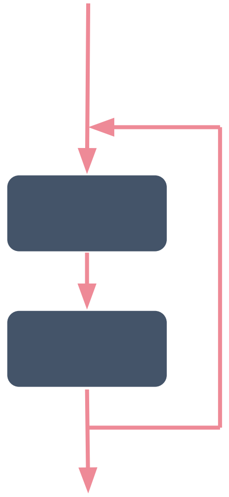
Control Flow



Sequence

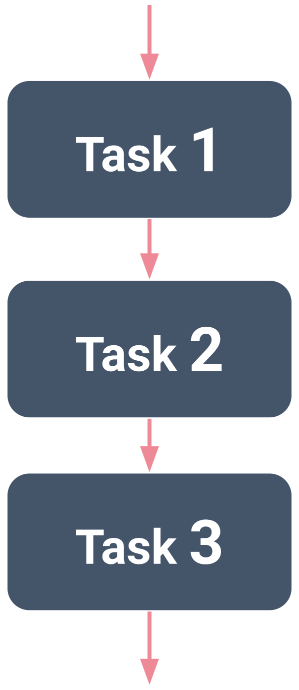


Selection



Iteration

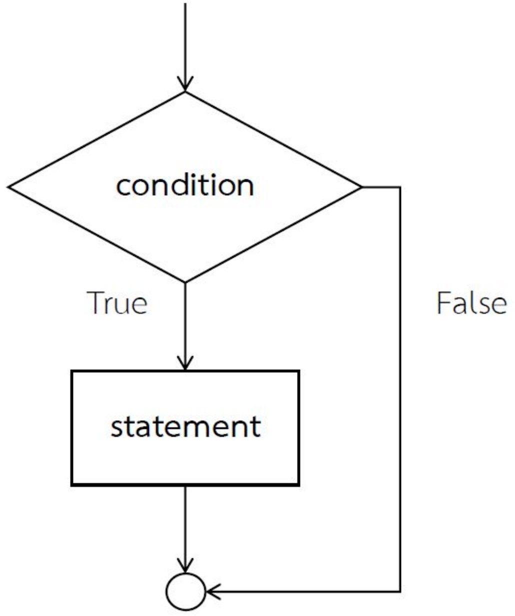
Sequential Structure



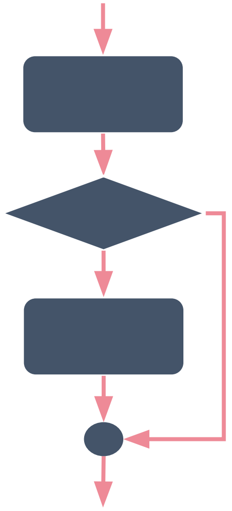
Sequence

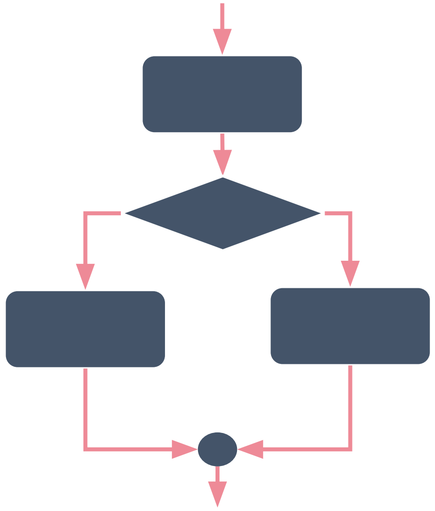
Selection Structure

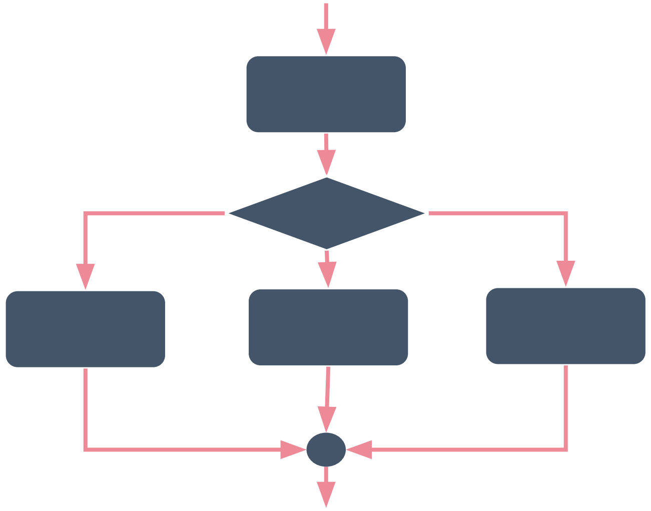
```
if <condition>:
    <statements>


x = 10
if x < 15:
    print("x is less than 15")
    print("OK")
print("Thank you")
```



Selection Structure

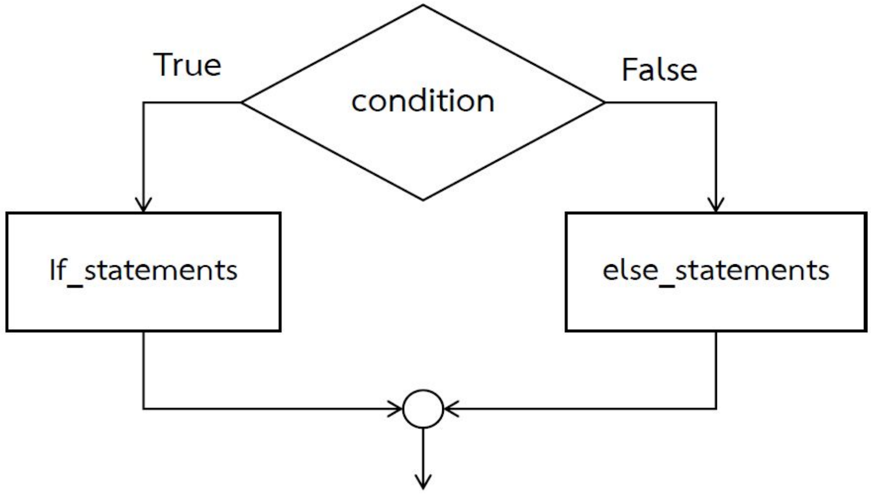

Single Selection


Double Selection


Multiple Selection

If/Else Statement

```
if <condition>:
        <if_statements>
else:
        <else_statements>
```



SCHOOL OF INTERNATIONAL & INTERDISCIPLINARY ENGINEERING PROGRAMS
King Mongkut's Institute of Technology Ladkrabang

If/Else Statement

```
money = 300
if money >= 350:
    print(‘You can buy this item’)
else:
    print(‘You don\’t have sufficient money to buy this bag’)
```

Selection: Even or Odd

Exercise 3

Write a program to calculate an area of a triangle by accepting a height and a width (base) of a triangle.

A program will have to verify that accepted inputs are positive numbers. Otherwise, it should display “Height and width should be positive numbers”

- Example input: triangle(10, 12)
- Expected output:
  - A triangular area is 60

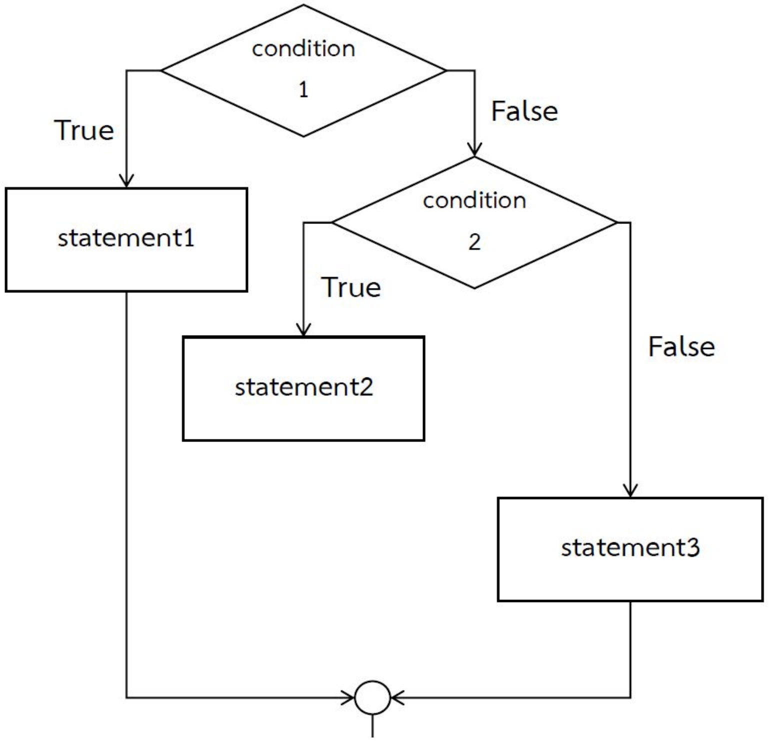
Example input: triangle(-5, 20)

Expected output:

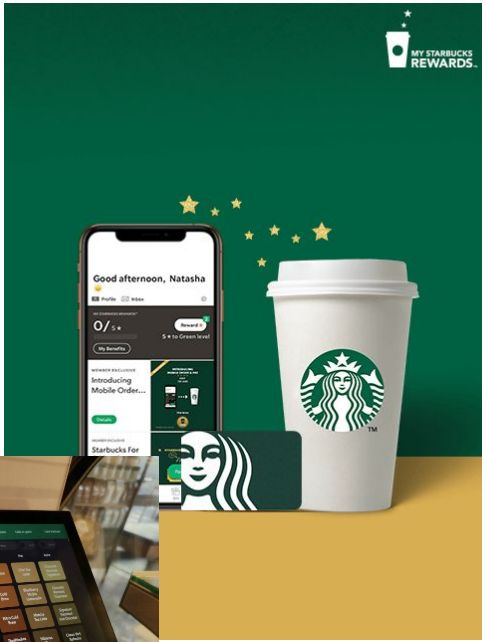
Height and width should be positive numbers

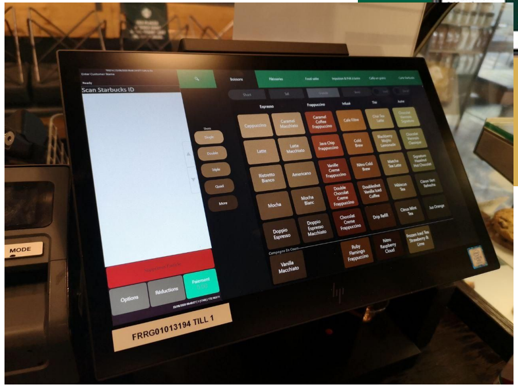
If-Elif Statement

```
if <condition1>:
        <statement_1>
elif <condition2>:
        <statement_2>
…
else:
        <statement_n>
```



Selection: Rewards



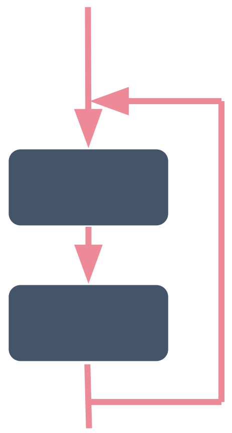


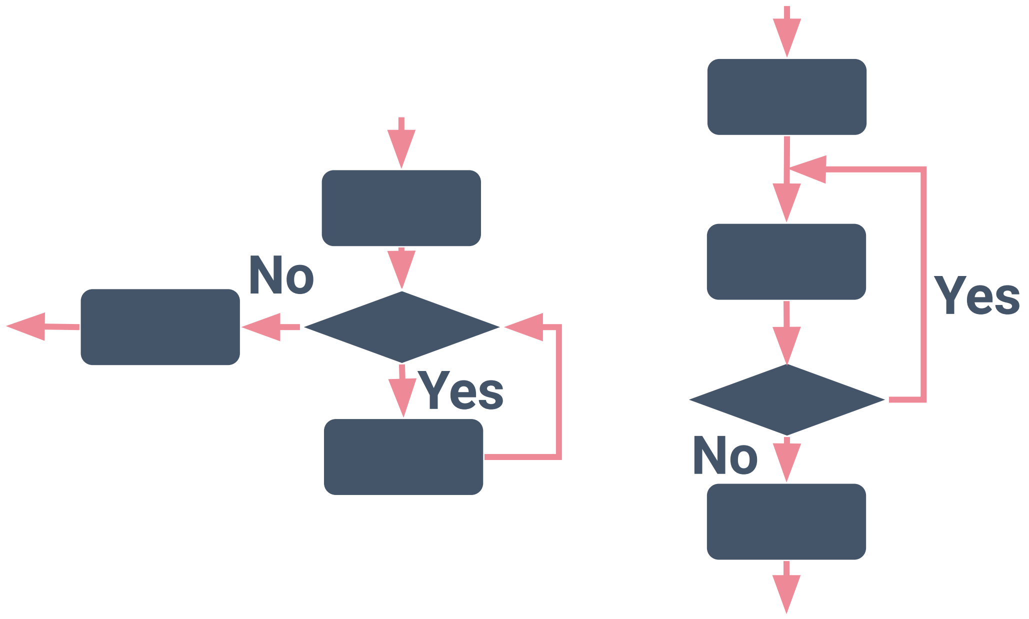
Exercise 4: Grading

Write a program to return an appropriate grade given a score according to the table below.

Grade	Score
A	80-100
B	70-79
F	0-69
Not in range	Outside 0-100

Iteration





Infinite Loop

Finite Loop

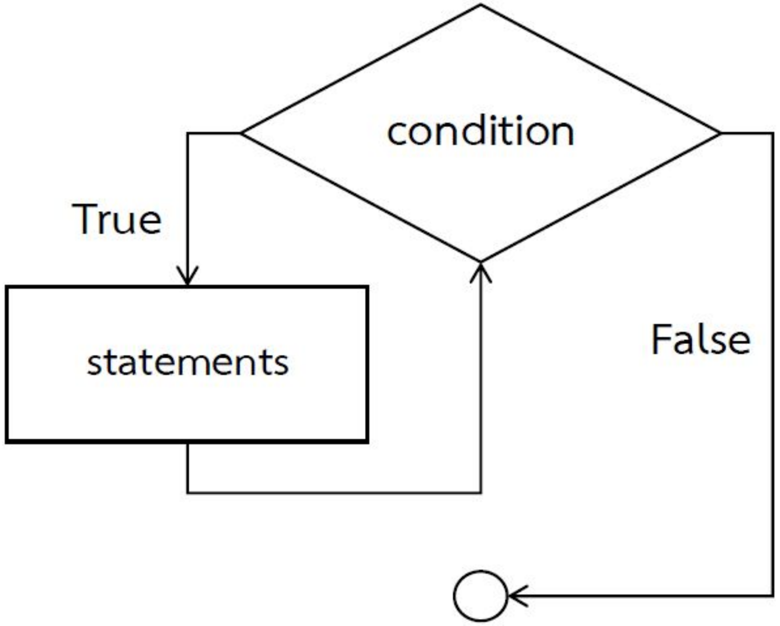
Repetition Structure

while loop

for loop

While Statement

```
while <condition>:
        <statements>


i = 1
while i <= 10:
        print(i, end = ‘, ‘)
        i = i + 1
```



Iteration: Input Range

A program that accepts input between 1 and 100 only.

Exercise 5: Grading

Modify the program in Exercise 4 so that it should iterate the input until -1 is given to stop the program and print “Thank you” message.

Grade	Score
A	80-100
B	70-79
F	0-69
Not in range	Outside 0-100

For Statement

```
for variableName in groupOfValues:
    <statements>
```

```
for x in range(1, 6):
    print(x, ‘squared is’, x*x)
```

range()

```
range(start, end, step)
```

```
list(range(10))

list(range(1, 11))

list(range(0, 30, 5))

list(range(0, -10, -1))
```

For Statement

```
for i in range(10, 0, -1):
	print(i, end= ‘, ’)


names = [‘Jane’, ‘John’, ‘Eric’, ‘Elon’]
for i in range(len(names):
	print(names[i], end = ‘, ‘)
```

Exercise 6: Control Flow

Write a flowchart describing the logic of factorial function which accepting a number as an input.

- Example
  - Enter factorial number: 6
  - The result is 720

Hint: The factorial n!, is the product of all positive integers less than or equal to n.
*For example: 5! = 5 * 4 * 3 * 2 * 1 = 120*

Exercise 6: Control Flow

Function

```
def function_name(args…):
    <statements>


def function_name(args…):
    <statements>
    return value
```

Function

```
def hello(name):
    print(“Hello”, name)


def area(width, height):
    c = width * height
    return c
```

Lab 1-1

Write a Python function **is_multiple(n, m)** that accepts two integer values (*n* and *m*) and returns a result as *True* or *False*.

- Return True if *n* is a multiple of *m*, that is, *n* = *m* * *i* for some integer *i*
- Return False otherwise

Example

- is_multiple(10, 3) -> False

Lab 1-2

Write a Python function **is_even(k)** that accepts an integer value (*k*) and returns a result as *True* or *False*.

- Return True if *k* is even
- Return False otherwise
- Multiplication, modulo, or division operators are not allowed.

Example

- is_even(20) -> True

Lab 1-3

Write a Python function **minmax(data)** that accepts a sequence of one or more numbers, and returns the smallest and largest numbers, in the form of a tuple of length two.

- Built-in functions min() or max() are not allowed

Example

- minmax([10, 50, 9, 5, 120, 18]) -> (5, 120)