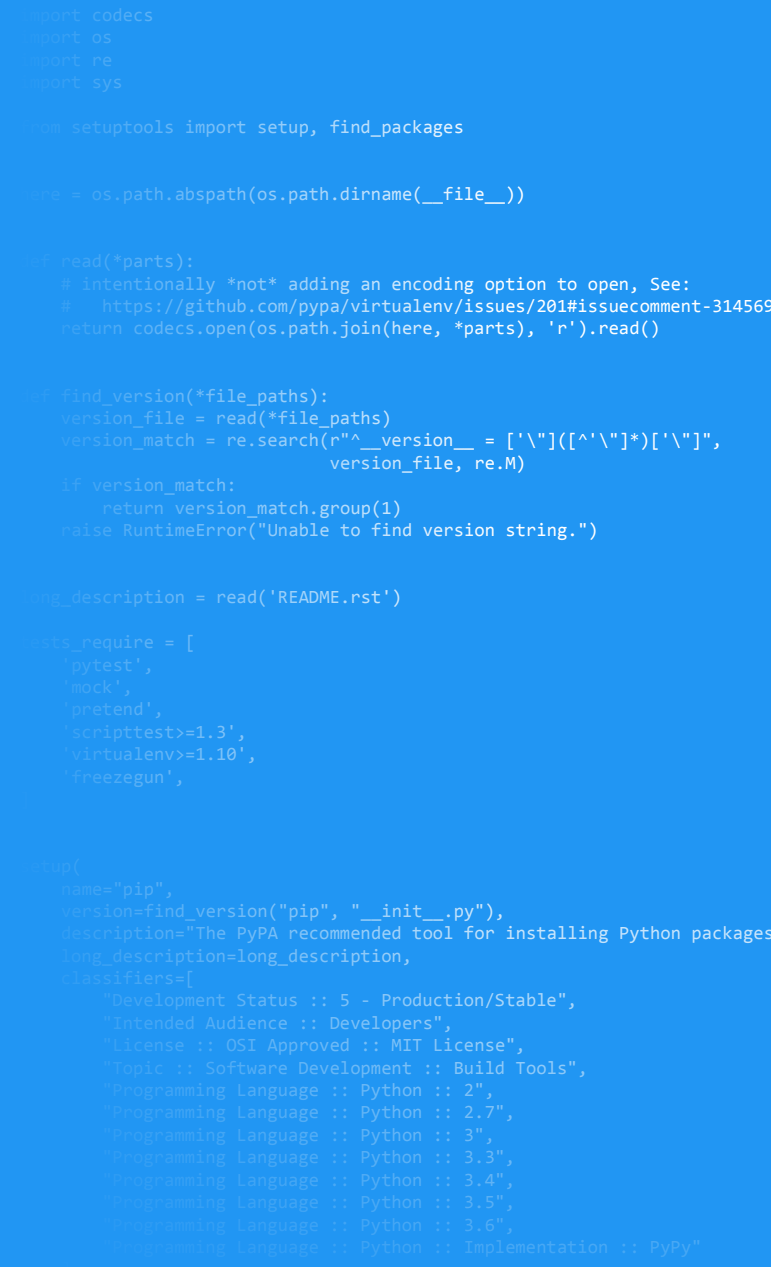
Expressions

Arithmetic and Boolean expressions

Control flows

01204111 Section 1

MIKE Laboratory



What Is an Expression?

- An ***expression*** is something that can be evaluated to a value.
  - An ***arithmetic expression*** can be evaluated to a numerical value.
- In our example, it is the part

```
82+64+90+75+33
```

- This part gets evaluated by the computer.
- The result is 344.

Warning – Operator Precedence

- In Python (and most programming languages), different operators have different precedence in order of operations.
- For example, `*` has precedence over `+` in this expression, no matter how many spaces are used.

```
82+64+90+75+33 * 0.8
```

- Therefore, the above expression is equivalent to:

```
82+64+90+75+(33*0.8)
```

- which is not as the code's intended purpose.

Operator Precedence

- Python (and most programming languages) evaluates expressions in this order.
- Operations of the same precedence are evaluated *from left to right (except **)*
- Parentheses decrease ambiguity. Use it when necessary.
- The operators which the sign is different from those in mathematics are
  - ** (Exponentiation)
  - % (remainder after division)
  - // (integer division)

Operators	Precedence
( )	Highest
**	:
* / // %	:
+ -	Lowest

Operator Precedence (Examples)

Expression	Equivalent to
2*3+4*5	(2*3)+(4*5)
1+2+3+4	((1+2)+3)+4
(2+3)/5*4	((2+3)/5)*4
3-2-5-(7+6)	((3-2)-5)-(7+6)
10+9%2+30	(10+(9%2))+30
10/2*5%3	((10/2)*5)%3
2**3**2	2**(3**2)

Task: *Average*

- Find the average of 82, 64, 90, 75 and 33

```
print('Summation: ')
print( 82+64+90+75+33 )
print('Average: ')
print(82+64+90+75+33 / 5)
```

```
print('Summation: ')
print( 82+64+90+75+33 )
print('Average: ')
print((82+64+90+75+33)/5)
```

Average – Revised Program

- We now store the result of the total amount in a ***variable***.
- Values stored in variables could be “referenced” anytime in the program.

```
sum = 82+64+90+75+33
print('Summation: ')
print(sum)
print('Average: ')
print(sum/5)
```

Integer vs. Floating Point Division

- // is integer division, when dividing two integers (whole numbers), the result is also a whole number.
  - The fraction part is discarded
- / is floating point division
- % is division remainder

Expression	Evaluated to
10/4	2.5
10//4	2 (not 2.5)
10//5	2 (not 2.0)
10/5	2.0
10%3	1

Task: *Sphere volume & surface area*

- Calculate the value of sphere volume and surface area
  - Sphere's volume: $(4/3)\ \pi\ r^3$
  - Sphere's surface area: $4\ \pi\ r^3$

```
pi = 3.14159265
radius = int(input('Radius: '))
print('Sphere\'s Volume: ')
print(4 / 3 * pi * radius ** 3)
print('Sphere\'s Surface area: ')
print(4 * pi * radius ** 3)
```

String operations

- We can't perform mathematical operations on strings, even if the strings look like numbers, so the following are illegal:
  - '2'-'1'
  - 'eggs'/'easy'
  - 'third'*'a charm'
- There are two exceptions, **+** and *****

```
>>> first = 'throat'
>>> second = 'warbler'
>>> first + second
    throatwarbler
>>> third = 'Spam' * 3
>>> print(third)
    SpamSpamSpam
```

+ for string concatenation

* for string repetition

Comments

```
'''
    weight(Newton) calculating program
    W = mg
'''
g = 9.8 # earth's gravitational constant in m/(s^2)
m = int(input('Mass: ')) # input m
print('Weight: %d Newton' , m * g)
```

Multiple lines Comment

Single line Comment

Useless Comment

Useful Comment

Python's Boolean Type

- ***bool*** have two possible values : **True** and **False**

```
>>> x = True
>>> y = False
>>> print(x)
True
>>> print(y)
False
>>> type(y)
<class 'bool'>
```

- Are **True** and **False** keywords?

Boolean Expression

- ***Boolean expression*** is an expression that give the value either **True** or **False**

```
>>> 5 > 3
True
>>> 5 <= 3
False
>>> x = 16
>>> x > 10 and x < 20
True
>>> not (x % 2 == 0)
False
```

```
>>> 'cat' != 'dog'
True
>>> pet = 'fish'
>>> nbOfFish = 12
>>> pet == 'hamster'
False
>>> pet == 'fish' and nbOfFish <= 10
False
```

Warning – Python is case-sensitive

- ***False*** and ***false*** are not the same.
- Also, ***True*** and ***true*** are not the same.
- Python's bool constants are written as **True** or **False**

```
>>> true = False
>>> true or False and True
False
```

Operators

Relational operators

equal	==
not equal	!=
greater than	>
greater than or equal	>=
less than	<
less than or equal	<=

Logical operators

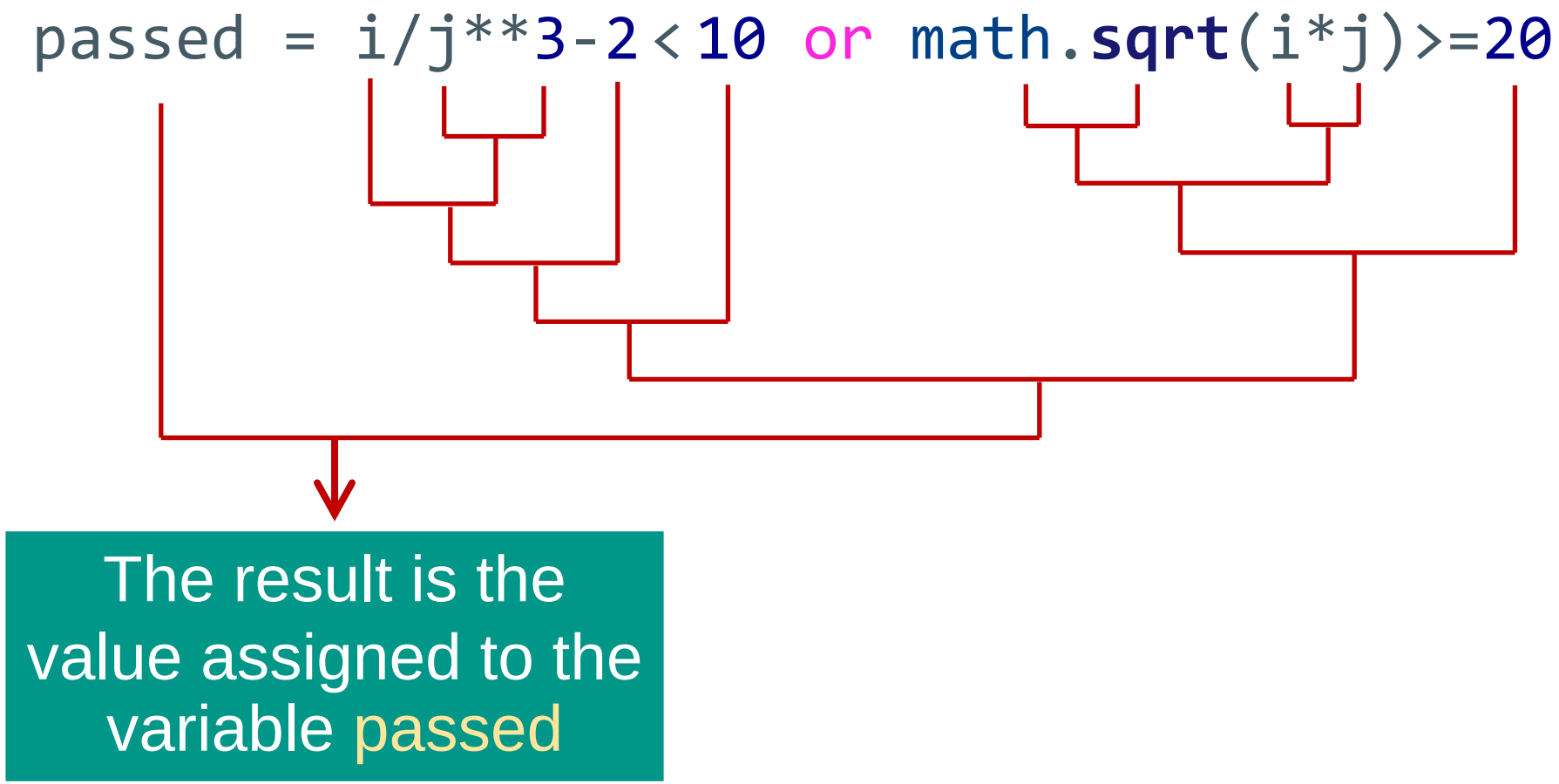
AND	**and**
OR	**or**
NOT	**not**

Operator Precedence

- From the highest precedence to the lowest down the table.
- Operators on the same row have the same precedence.

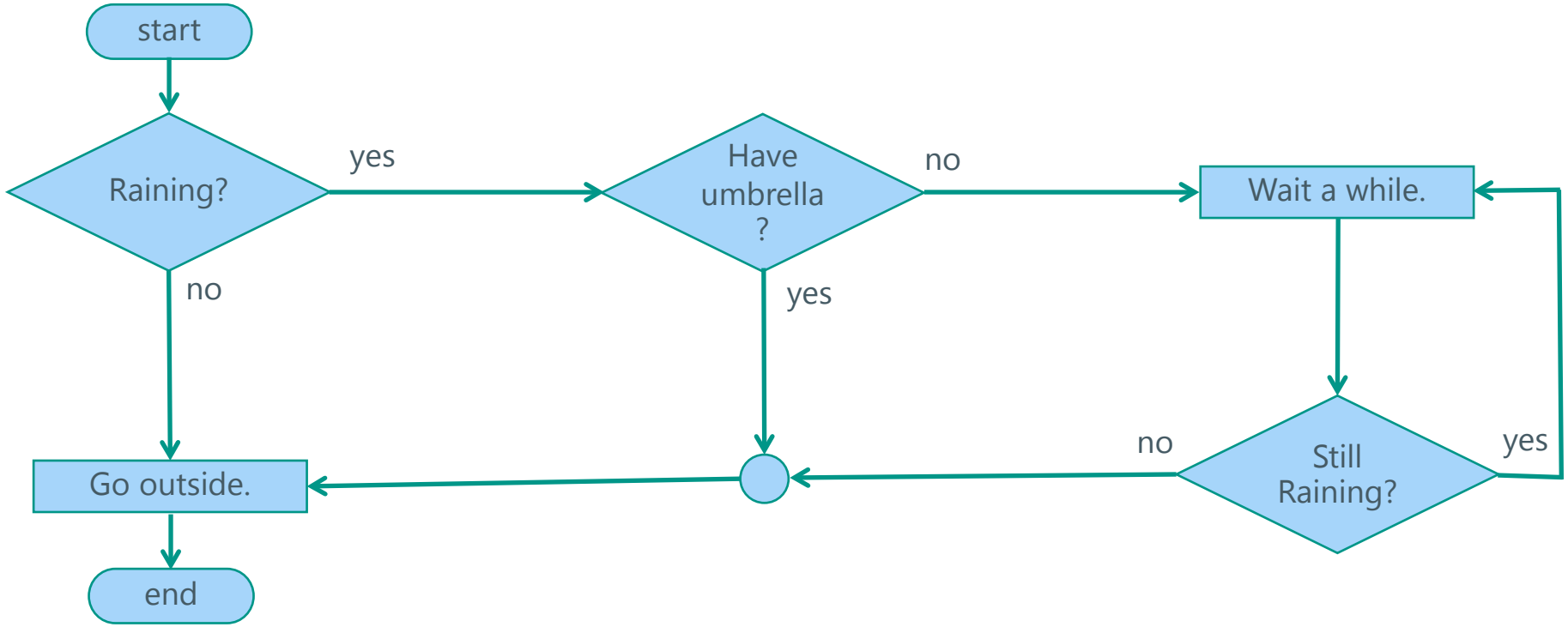
Category	Operators	Associativity
Subscription, call, attribute	a[x] f(x) x.attribute	left to right
Exponentiation	**	right to left
Unary sign	+x -x	left to right
Multiplicative	* / // %	left to right
Additive	+ -	left to right
Relational (comparison)	== != < > <= >=	left to right
Boolean NOT	not	left to right
Boolean AND	and	left to right
Boolean OR	or	left to right

Operator Precedence: Examples



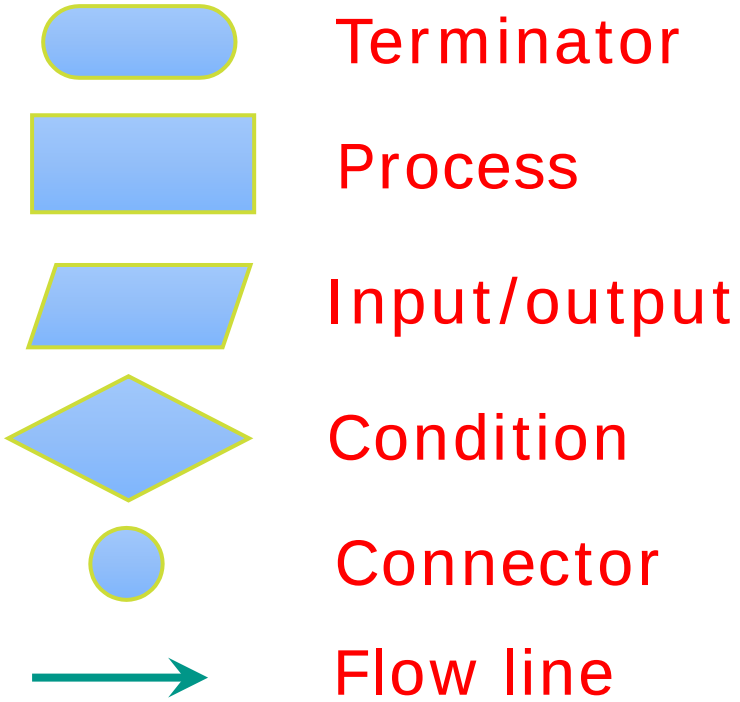
Flow control?

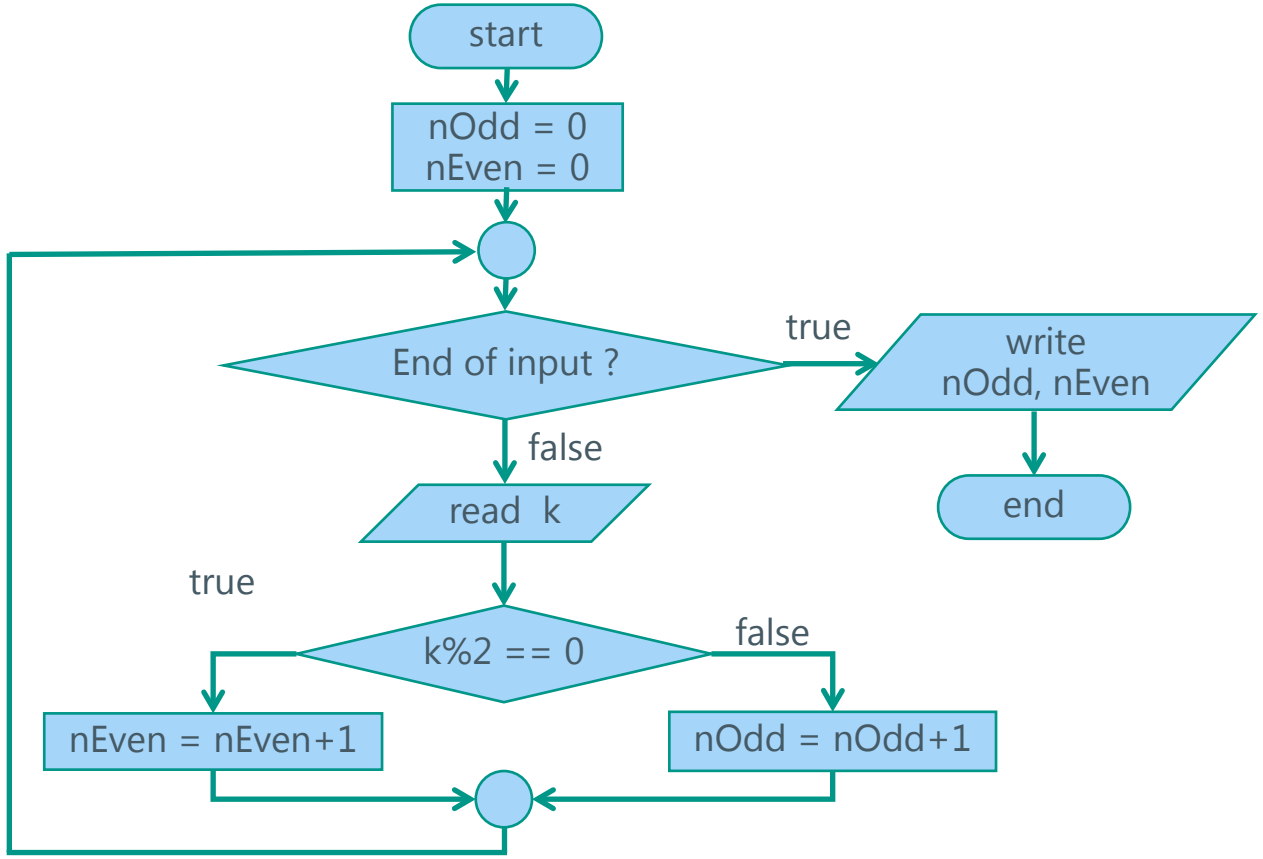
- What to do if it is raining?



Flowchart

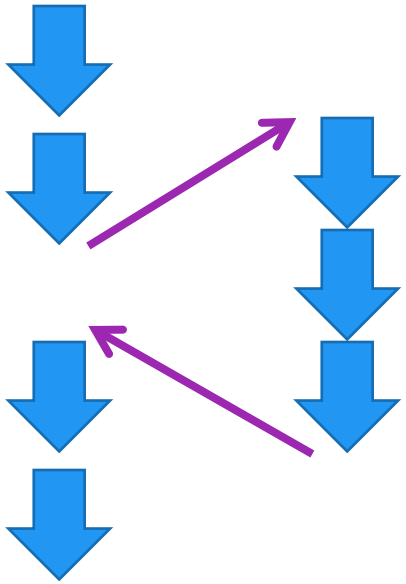
- ***Flow chart*** is a graphical representation of Controls



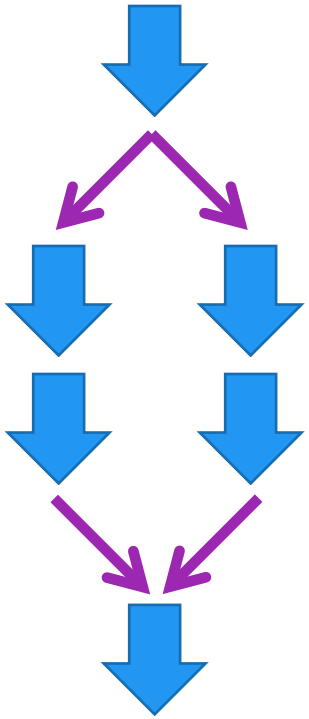


Schematic View of Flow Controls

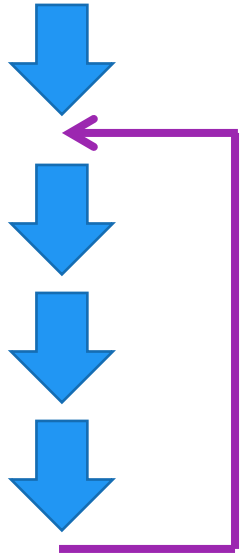
Sequence



Subroutine



Selection

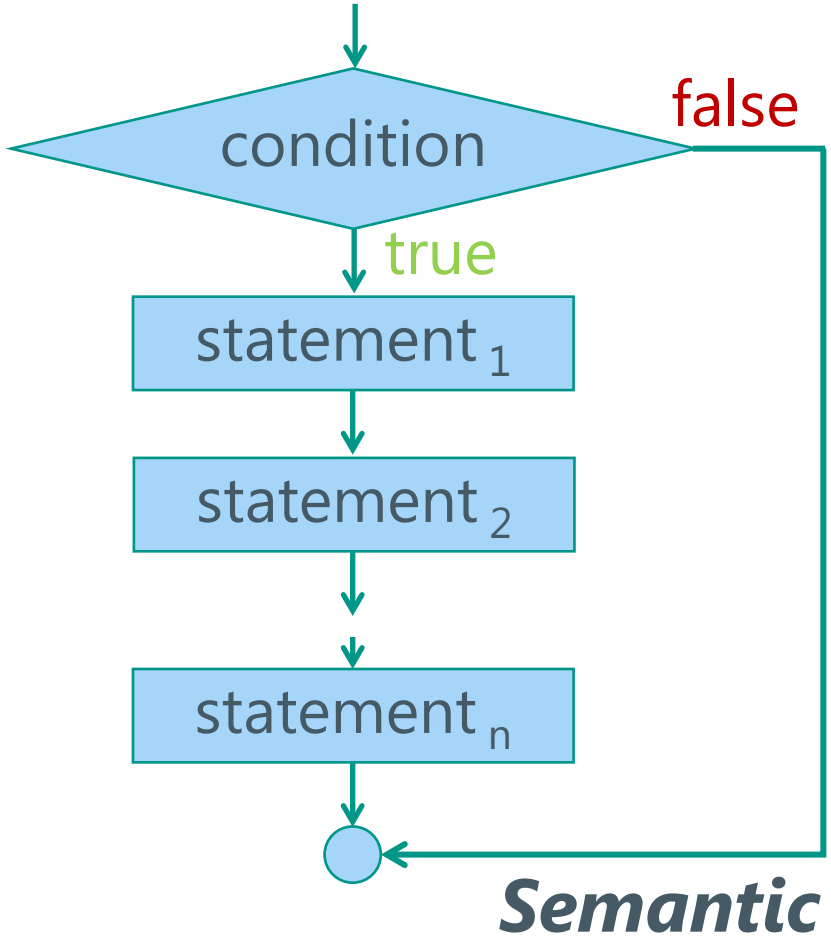


Repetition

if statement

```
if condition :
    statement1
    statement2
    …
    statementn
```

Syntax

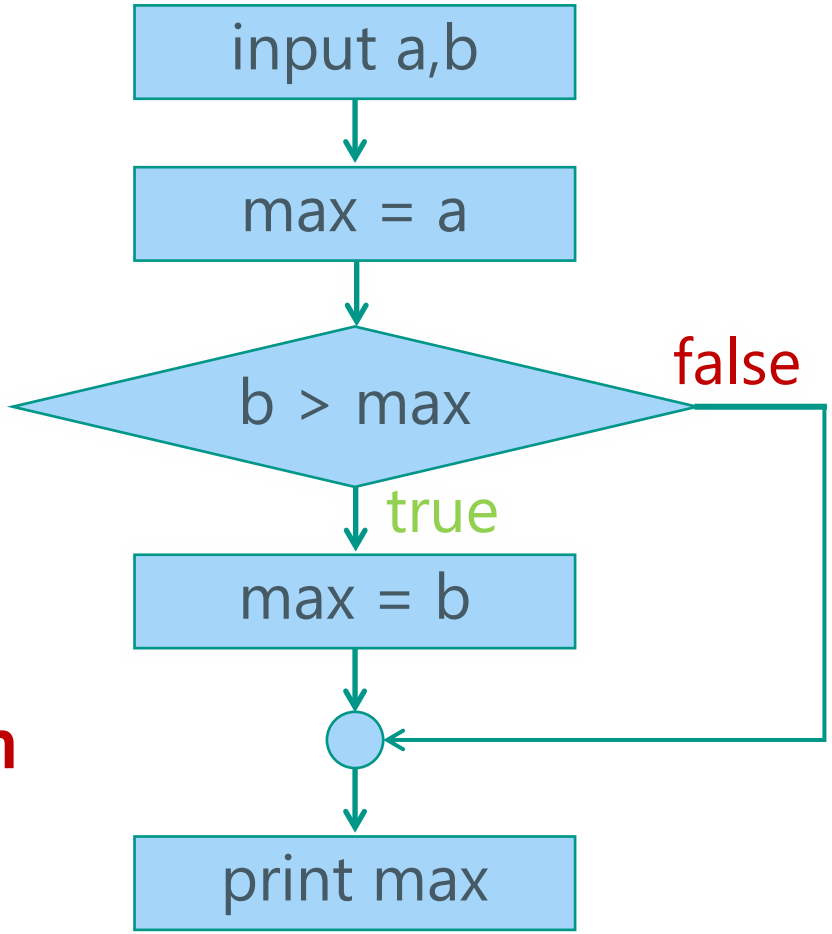


Semantic

Find max of two integers

```
a = int(input('enter 1st integer: '))
b = int(input('enter 2nd integer: '))
max = a
if b > max :
     max = b
print(max)
```

Note: Be Careful about indentation



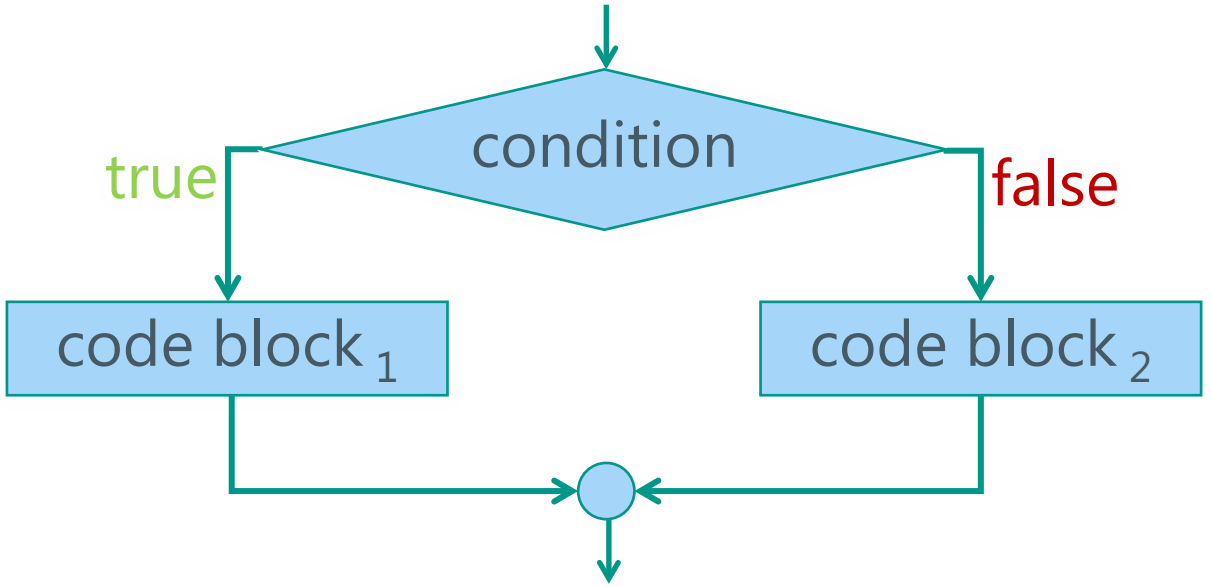
Find max of three integers

```
a = int(input('enter 1st integer: '))
b = int(input('enter 2nd integer: '))
b = int(input('enter 3rd integer: '))
max = a
if b > max :
    max = b
if c > max :
    max = c
print(max)
```

Write your own flowchart..

if-else statement

```
if condition :
      code block1
else :
      code block2
```



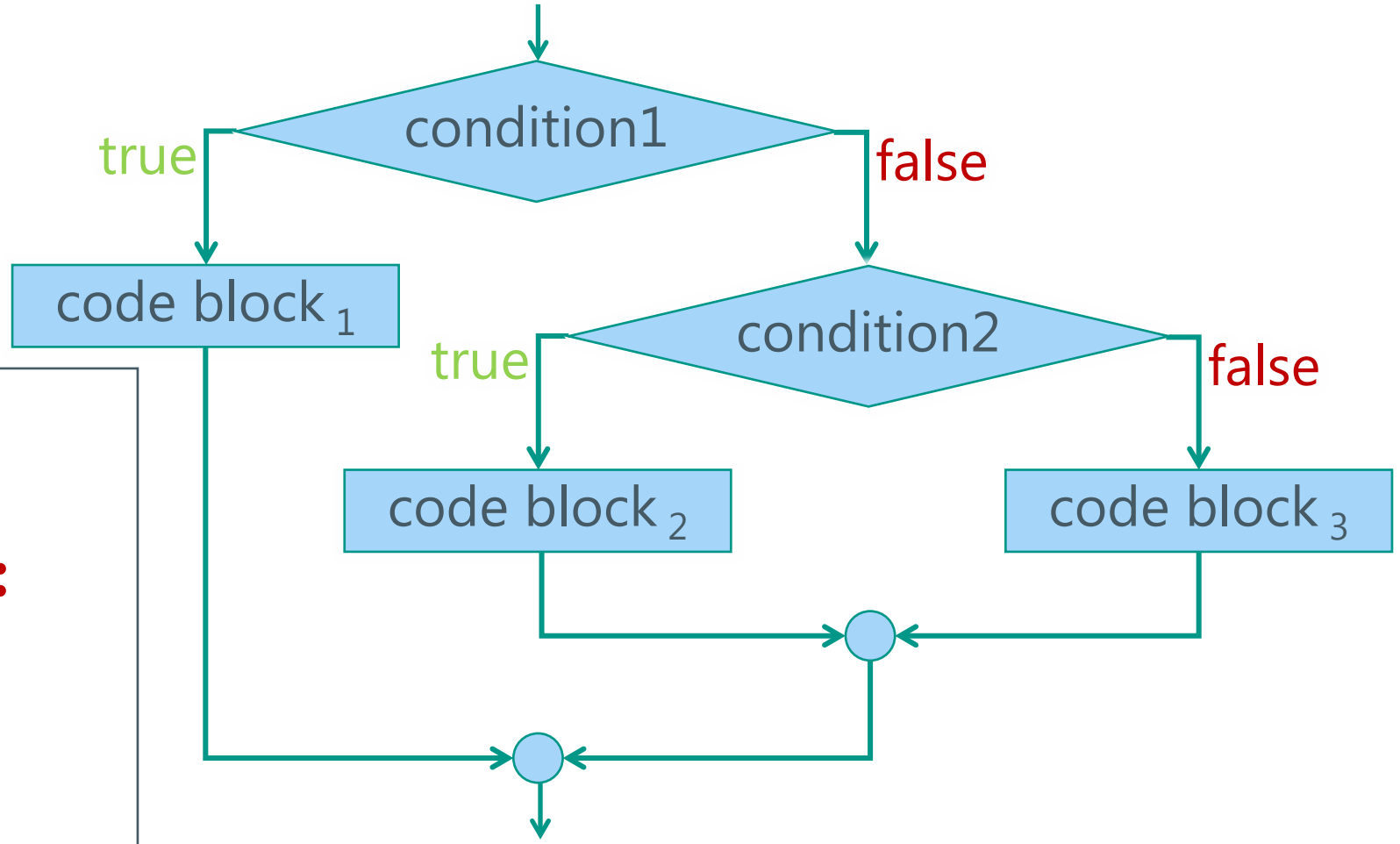
Odd or Even

```
x = int(input('enter 1st integer: '))
if x % 2 == 0 :
    print('%d is even number' %(x))
else :
    print('%d is odd number' %(x))
```

Write your own flowchart..

if-elif-else statement

```
if condition1 :
      code block1
elif condition2 :
      code block2
else :
      code block3
```



Positive Negative or Zero

```
x = int(input('enter 1st integer: '))
if x > 0 :
     print('%d is positive number' %(x))
elif x < 0 :
     print('%d is negative number' %(x))
else :
     print('%d equal zero' %(x))
```

Write your own flowchart..

Task: *electricity bill*

- Given the electricity unit charges and calculate total electricity bill according to the given condition:

For First 5 unit	**0 baht/unit**
For Next 10 unit	**0.50 baht/unit**
For Next 65 unit	**1.00 baht/unit**
For Next 120 unit	**1.50 baht/unit**
For unit above 200	**2.50 baht/unit**

Task: *electricity bill*

```
unit = int(input('input units: '))
if unit <= 5 :
      print(unit*0)
elif unit <= 15 :
      print(5*0 + (unit-5)*0.5)
elif unit <= 80 :
      print(5*0 + 10*0.5 + (unit-15)*1.0)
elif unit <= 200 :
      print(5*0 + 10*0.5 + 65*1.0 + (unit-80)*1.5)
else :
      print(5*0 + 10*0.5 + 65*1.0 + 120*1.5 + (unit-200)*2.5)
```

Note: Too much constants

Task: *electricity bill*

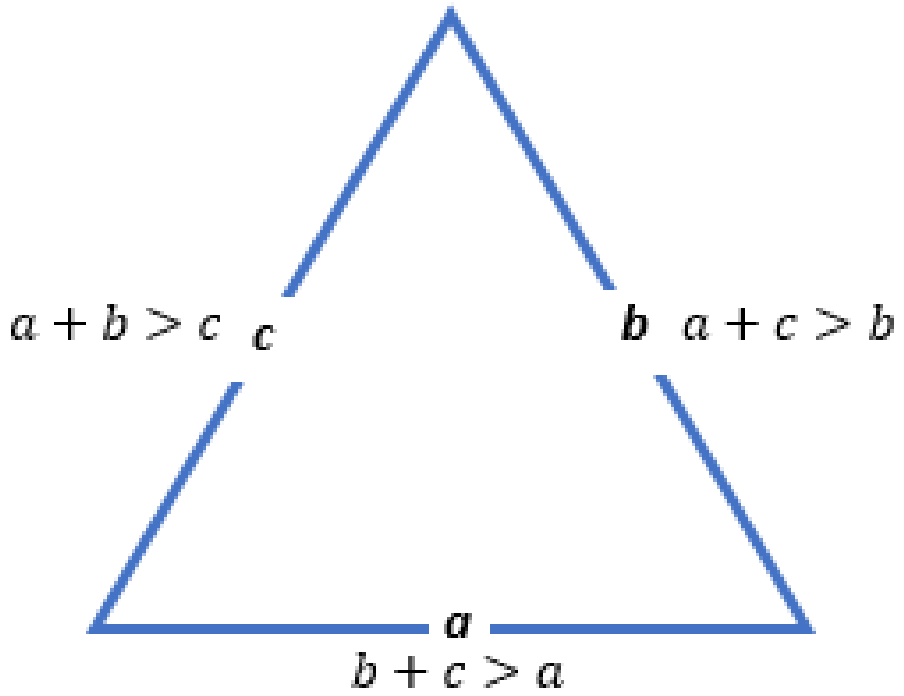
```
unit = int(input('input units: '))
cost = 0
if unit > 0 :
        unit -= 5
if unit > 0 :
        if unit < 10 :
                cost += unit*0.5
        else :
                cost += 10*0.5
        unit -= 10
```

```
if unit > 0 :
        if unit < 65 :
                cost += unit*1
        else :
                cost += 65*1
        unit -= 65
if unit > 0 :
        if unit < 120 :
                cost += unit*1.5
        else :
                cost += 120*1.5
        unit -= 120
if unit > 0 :
        cost += unit*2.5
print(cost)
```

Task: *is triangle valid*

- Given the three side of triangle then answering that triangle is valid or not

- Property of Triangle



Task: *is triangle valid*

```
a = int(input('input 1st side: '))
b = int(input('input 2nd side: '))
c = int(input('input 3rd side: '))
if x a+b > c :
      if x a+c > b :
            if x b+c > a :
                  print('triangle is valid')
            else :
                  print('triangle is not valid')
      else :
            print('triangle is not valid')
else :
      print('triangle is not valid')
```

```
if x a+b > c and x a+c > b and x b+c > a :
      print('triangle is valid')
else :
      print('triangle is not valid')
```

Task: *Solving quadratic equations*

- Given the three coefficients ***a***, ***b*** and ***c*** of quadratic equation ***$ax^2 + bx + c = 0$*** where a ≠ 0, find the **roots** of the equation.

- You can find **roots** from $$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- This equation gives two real roots when ***$b^2 - 4ac > 0$***
  one real root when ***$b^2 - 4ac = 0$***
  two complex root when ***$b^2 - 4ac < 0$***

Task: *Solving quadratic equations*

INPUT:

```
Enter 1st coefficient: 0
Enter 2nd coefficient: -2
Enter 3rd coefficient: 5
```

OUTPUT:

```
Error: 1st coefficient cannot equal zero
```

```
Enter 1st coefficient: 1
Enter 2nd coefficient: 8
Enter 3rd coefficient: 16
```

```
Only one real root: -4.0
```

```
Enter 1st coefficient: 2
Enter 2nd coefficient: -1
Enter 3rd coefficient: -1
```

```
Two real roots: 1.0 and -0.5
```

```
Enter 1st coefficient: 5
Enter 2nd coefficient: 2
Enter 3rd coefficient: 1
```

```
No real root
```

Task: *Solving quadratic equations*

```
a = int(input('Enter 1st coefficient: '))
b = int(input('Enter 2nd coefficient: '))
c = int(input('Enter 3rd coefficient: '))
if a != 0 :
        D = (b * b) - (4 * a * c)
        if D < 0 :
                print('No real root')
        elif D == 0 :
                root = -b / (2*a)
                print('Only one real root: %.1f' %(root))
        else :
                root1 = (-b + (D**0.5))/ (2*a)
                root2 = (-b – (D**0.5))/ (2*a)
                print('Two real roots: %.1f and %.1f' %(root1,root2))
else :
        print('Error: 1st coefficient cannot equal zero')
```

Task: *Solving quadratic equations 2*

INPUT:

```
Enter 1st coefficient: 0
Enter 2nd coefficient: -2
Enter 3rd coefficient: 5
```

OUTPUT:

```
Error: 1st coefficient cannot equal zero
```

```
Enter 1st coefficient: 1
Enter 2nd coefficient: 8
Enter 3rd coefficient: 16
```

```
Only one real root: -4.0
```

```
Enter 1st coefficient: 2
Enter 2nd coefficient: -1
Enter 3rd coefficient: -1
```

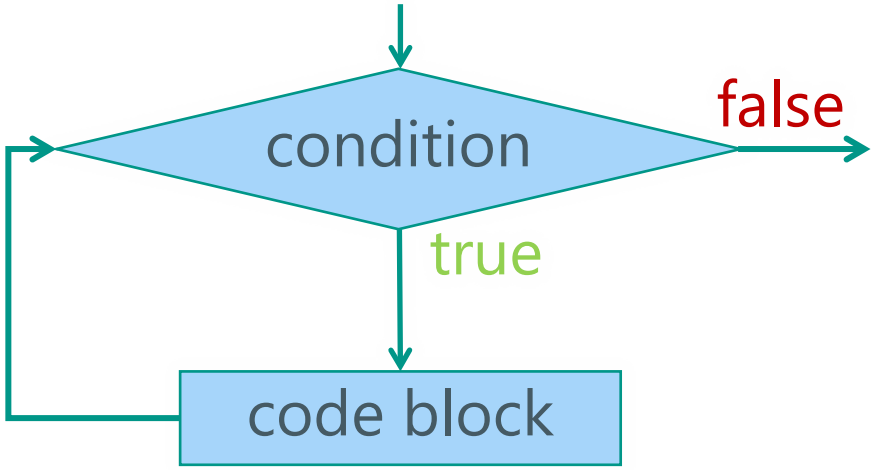
```
Two real roots: 1.0 and -0.5
```

```
Enter 1st coefficient: 5
Enter 2nd coefficient: 2
Enter 3rd coefficient: 1
```

```
Two complex roots: -0.2+0.4i and -0.2-0.4i
```

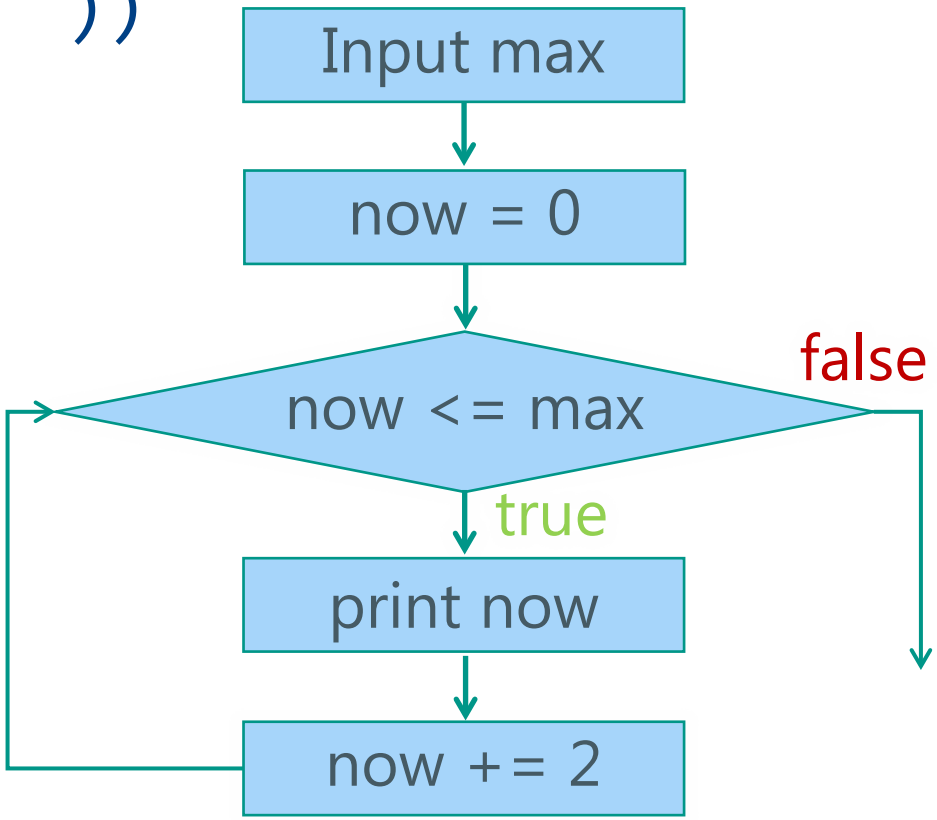
while statement

```
while condition :
    code block
```



Even numbers [0 2 4 6 8 10 12 ...]

```
max = int(input('enter maximum: '))
now = 0
while now <= max :
      print(now, end=' ')
      now += 2
```

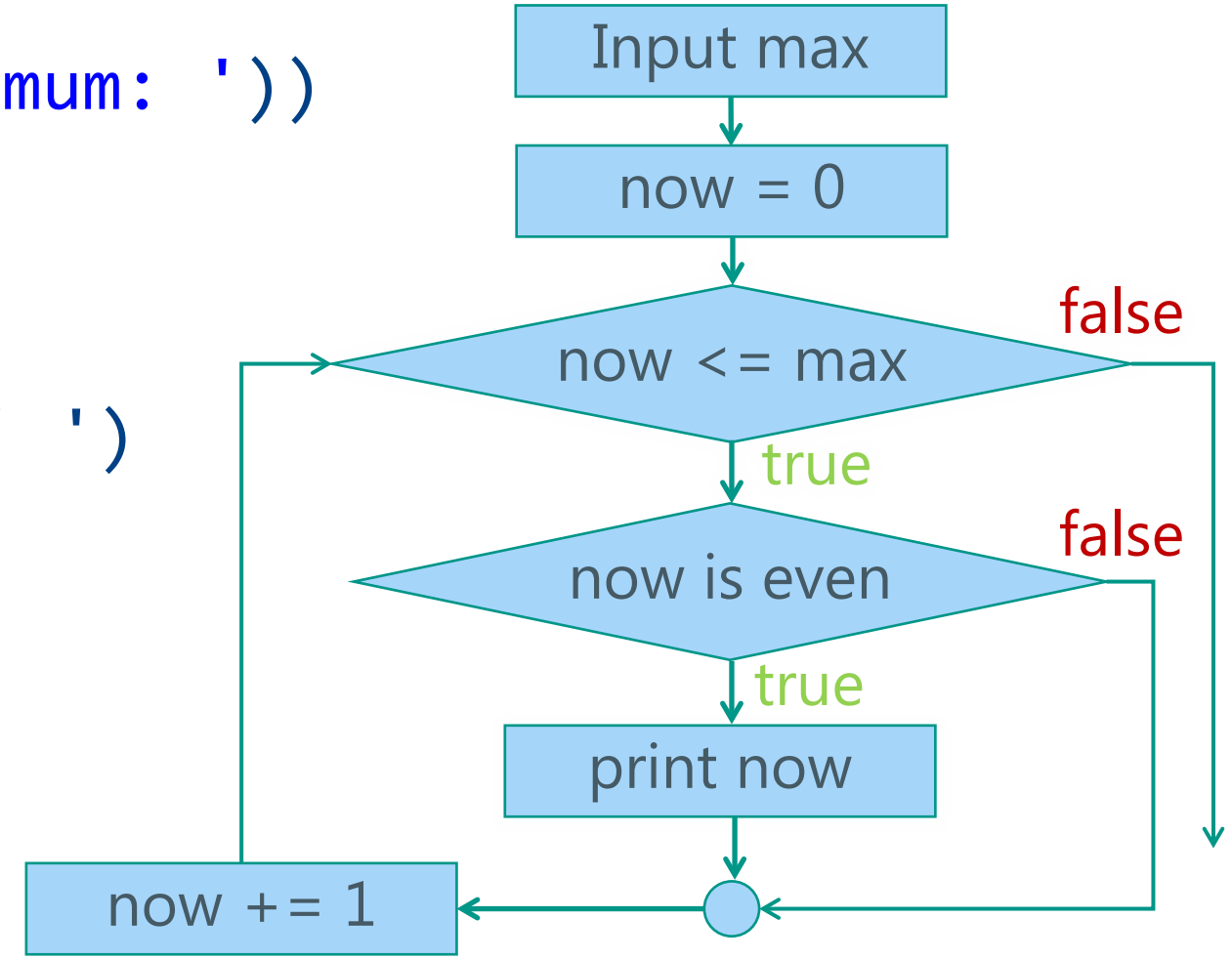


Even numbers [0 2 4 6 8 10 12 ...]

```
max = int(input('enter maximum: '))
now = 0
while now <= max :
    if now % 2 == 0 :
        print(now, end = ' ')
    now += 1
```

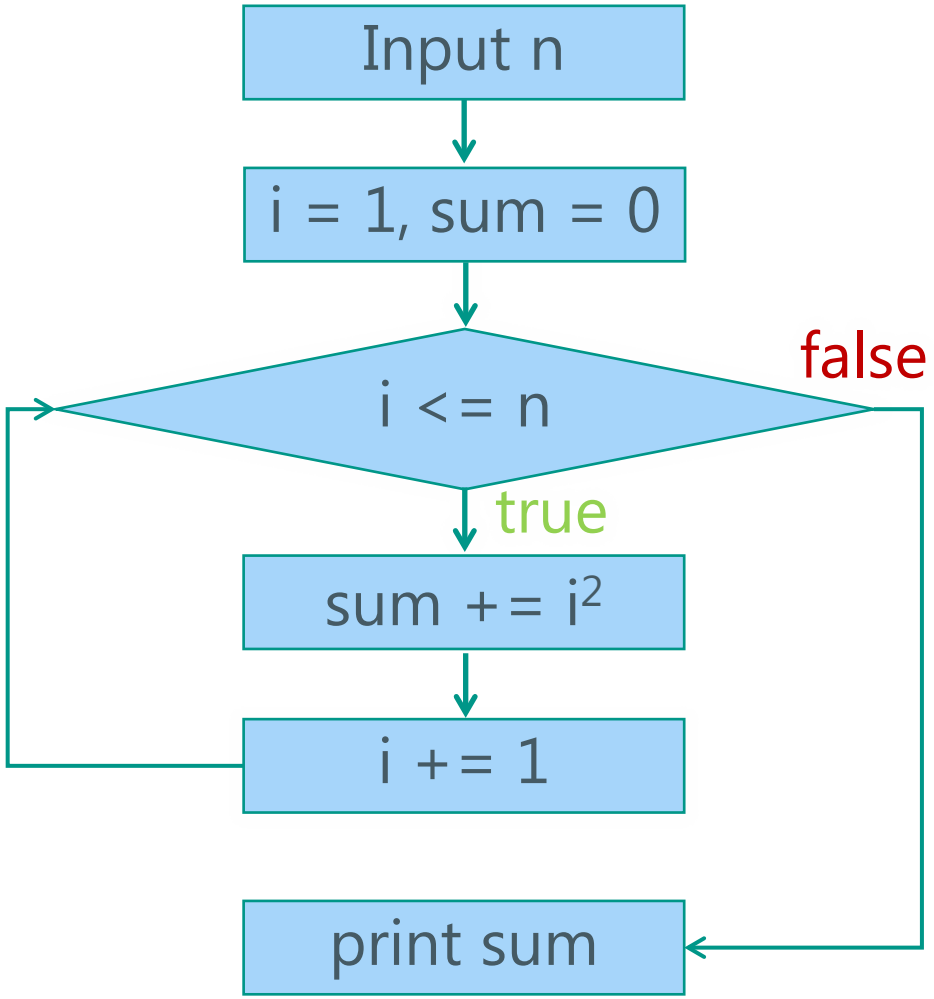
```
while now <= max/2 :
    print(now*2, end=' ')
    now += 1
```

Write your own flow chart..



Sum of $1^2,2^2,3^2,\ldots,n^2$

```
n = int(input('enter max: '))
i = 1, sum = 0
while i <= n :
      sum += i**2
      i += 1
print(sum)
```



Fibonacci [1,1,2,3,5,8,13,21,34,55,...]

```
maxTerm = int(input('enter maximum term: '))
nowTerm = 0, nowVal = 0, lastVal = 1
while nowTerm <= maxTerm :
    tmp = nowVal
    nowVal += lastVal
    lastVal = tmp
    print(nowVal, end=',')
    nowTerm += 1
```

f(x) = f(x-1) + f(x-2) ; f(0)=f(1)=1

x is positive integer

Task: *Prime or not*

- **Prime number** is a whole number greater than 1 whose only two whole number factors are 1 and itself
- Input an integer then tell that it is a prime number or not

Task: *Prime or not*

```
x = int(input('Enter 1st coefficient: '))
if x > 1 :
      check = True, i = 2
      while i < x :
            if x % i == 0 :
                  check = False
            i += 1
      if check :
            print('%d is prime number' %(x))
      else :
            print('%d is not prime number' %(x))
else :
      print('%d is not prime number' %(x))
```

Task: *Sum until zero*

- Write a program that ask for a number until input zero then show the summation of all input with 2 decimal places

INPUT:

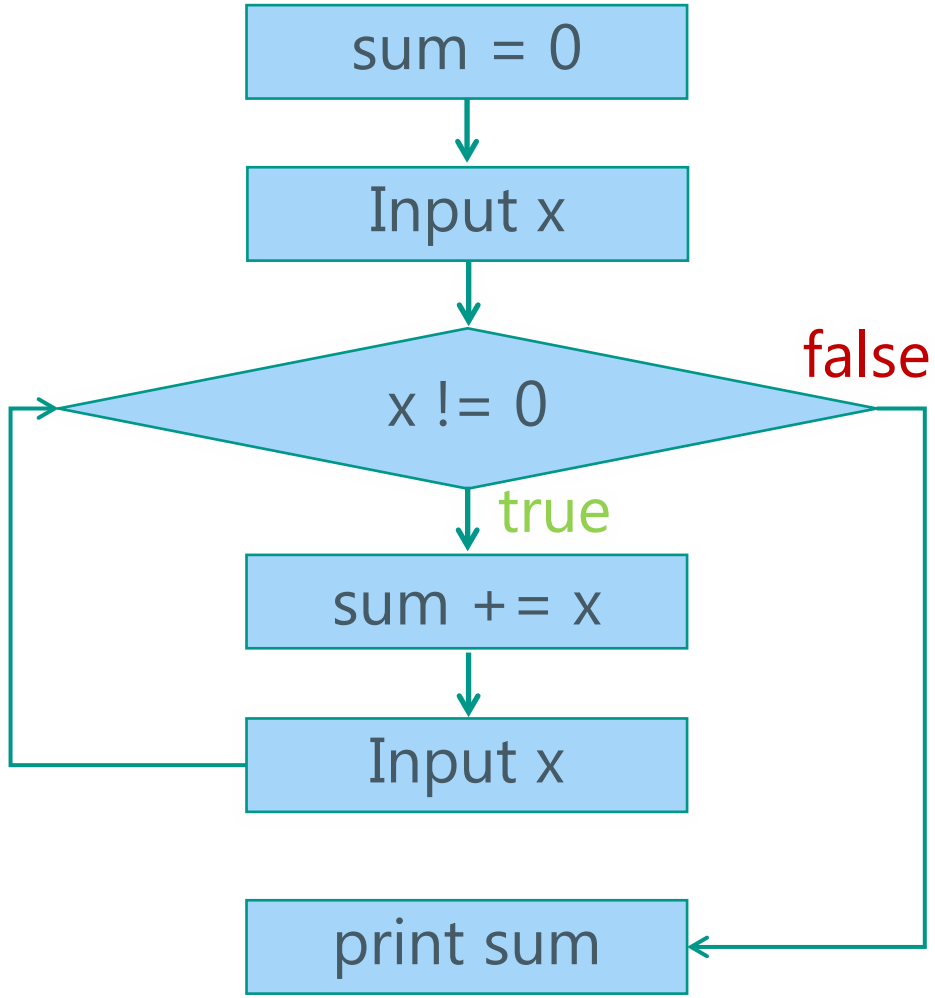
```
Enter a number: 1.5
Enter a number: 2.4
Enter a number: -3.4
Enter a number: 4.2
Enter a number: 0
```

OUTPUT:

```
Sum = 4.70
```

Task: *Sum until zero*

```
sum = 0
x = float(input('Enter a number: '))
while x != 0 :
      sum += x
      x = float(input('Enter a number: '))
print('Sum = %.2f' %(sum))
```



What will happen with this program?

```
x = 5, y = 0
while y <= x :
    print(x)
    x -= 1
```

```
5
4
3
2
1
0
```

```
x = 5, y = 0
while y <= x :
    print(x)
    #x -= 1
```

```
5
5
5
5
5
5
...
```

Infinite loop / Endless loop
a sequence of instructions which having no terminating condition, or having condition that can never be met

Infinite loop

Other Example

```
while True :
    print('Hello')
print('World')
```

To Exit infinite loops
- CTRL – C
- make a terminate condition
- break statement

```
x = 10
while x > 0 :
    print(x)
    x += 1
```

```
x = 0
while x % 2 == 0 :
    print(x)
    x += 2
```

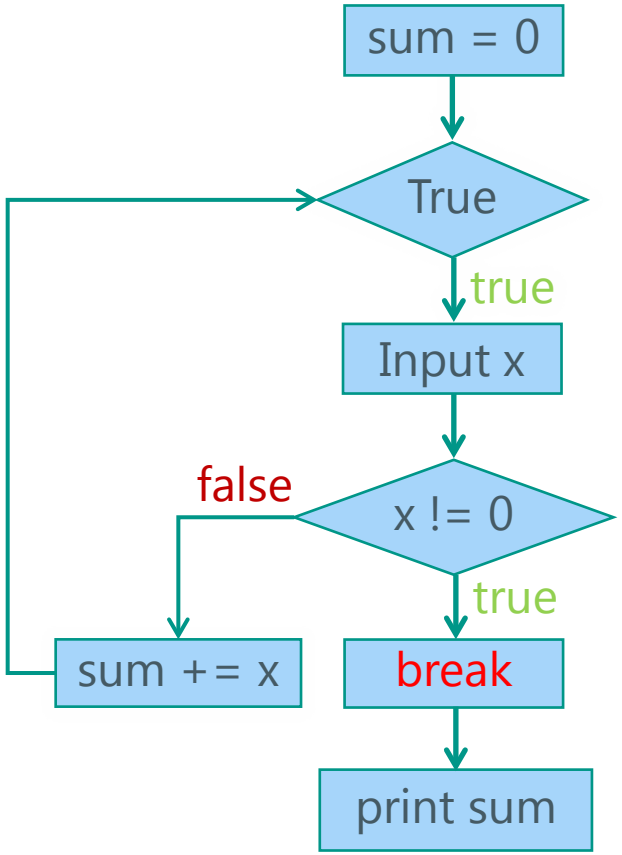
```
x = 0
while x != 10 :
    print(x)
    x += 3
```

break statements

- If the execution reaches a **break** statement, it immediately exits the while loop's clause.

```
sum = 0
while True :
      x = float(input('Enter a number: '))
      if x == 0 :
            break
      sum += x
print('Sum = %.2f' %(sum))
```

Sum until Zero (new Version)

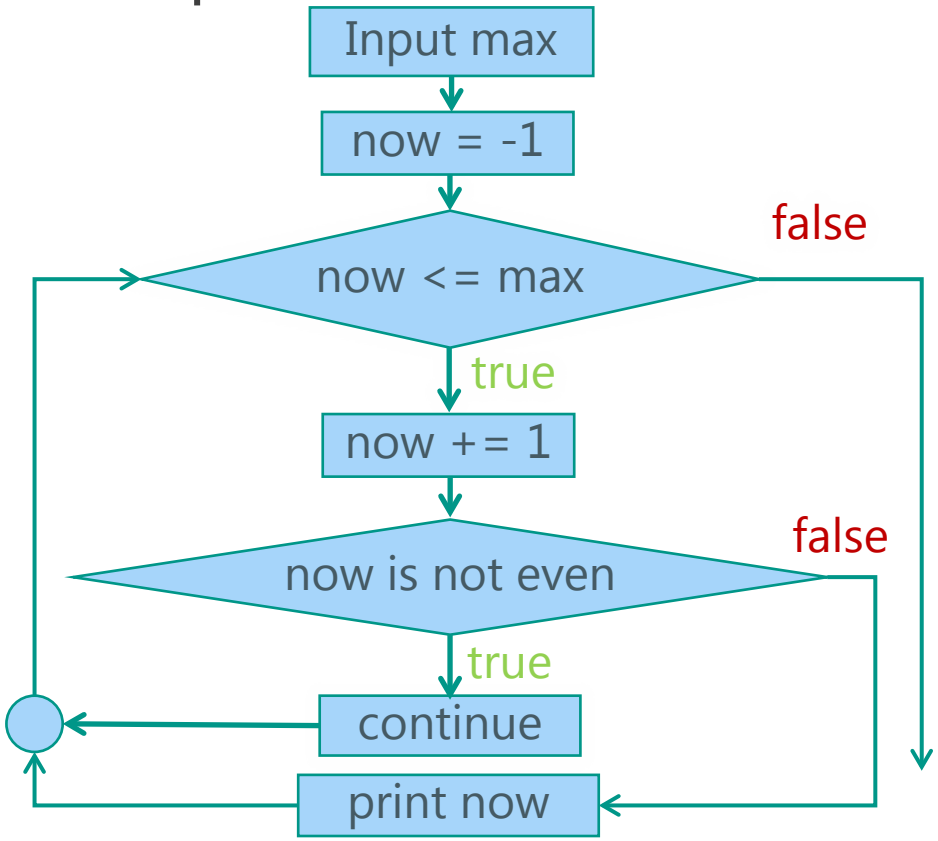


continue statements

- When the execution reaches a **continue** statement, the execution immediately jumps back to the start of the loop and revaluates the loop's condition.

```
max = int(input('enter maximum: '))
now = -1
while now <= max :
      now += 1
      if now % 2 != 0 :
            continue
      print(now, end = ' ')
```

Even Numbers (new Version)



Task: *Triangle star*

- Input a positive integer then draw the triangle

INPUT:

```
Enter a size: 3
```

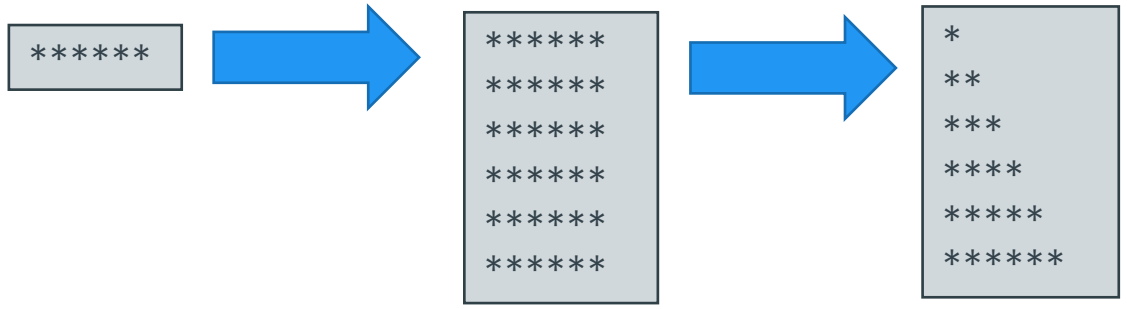
OUTPUT:

```
*
**
***
```

```
Enter a size: 7
```

```
*
**
***
****
*****
******
*******
```

Hint: try to implement code following these steps when the input is 6



Task: *Triangle star*

```
size = int(input('Enter size: '))
col = 0
while col < size :
      print('*', end='')
      col += 1
print()
```

OUTPUT:

```
******
```

Task: *Triangle star*

```
size = int(input('Enter size: '))
row = 0
while row < size :
      col = 0
      while col < size :
            print('*', end='')
            col += 1
      print()
      row += 1
```

OUTPUT:

```
******
******
******
******
******
******
```

Task: *Triangle star*

```
size = int(input('Enter size: '))
row = 0
while row < size :
      col = 0
      while col < size :
            if col <= row :
                  print('*', end='')
            col += 1
      print()
      row += 1
```

OUTPUT:

```
*
**
***
****
*****
******
```

Task: *Triangle star*

```
size = int(input('Enter size: '))
row = 0
while row < size :
      col = 0
      while col <= row :
            print('*', end='')
            col += 1
      print()
      row += 1
```

OUTPUT:

```
*
**
***
****
*****
******
```

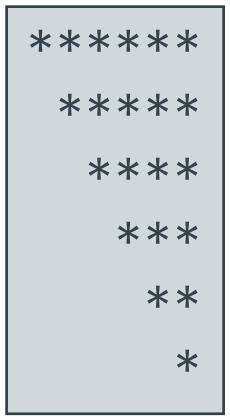
Task: *Triangle star*

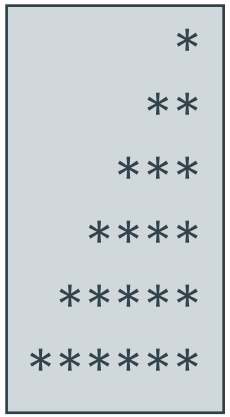
```
size = int(input('Enter size: '))
row = 0
while row < size :
      col = row
      while col < size :
            print('*', end='')
            col += 1
      print()
      row += 1
```

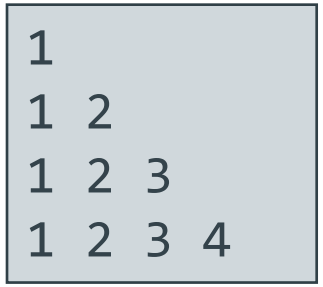
OUTPUT:

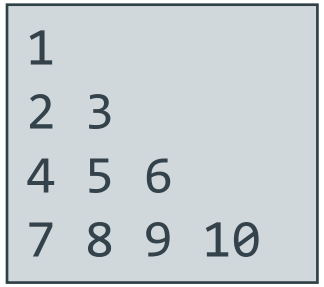
```
******
*****
****
***
**
*
```

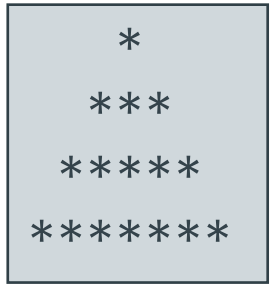
Now you can make these...

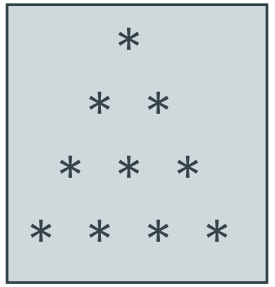


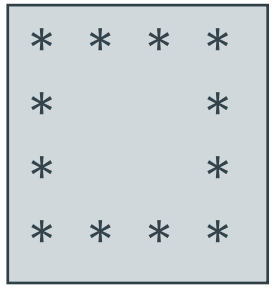


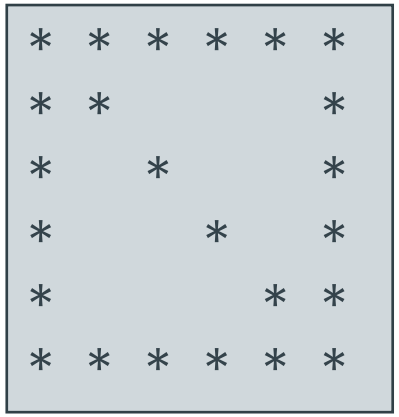


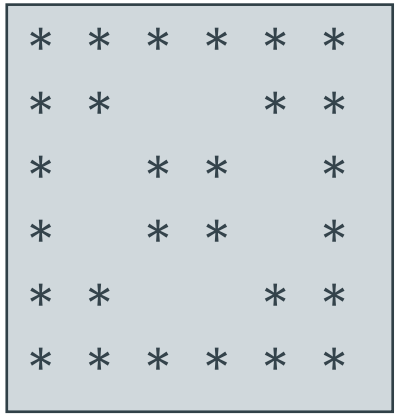


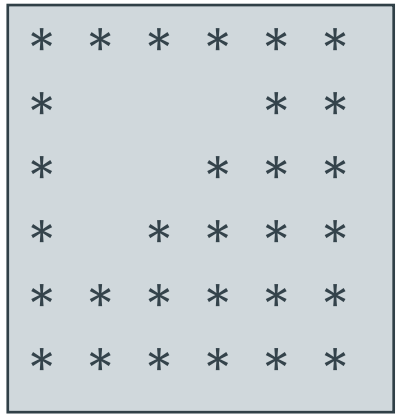


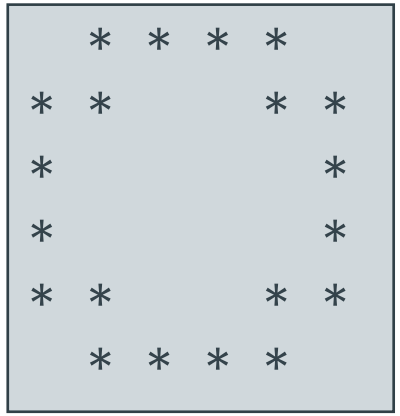


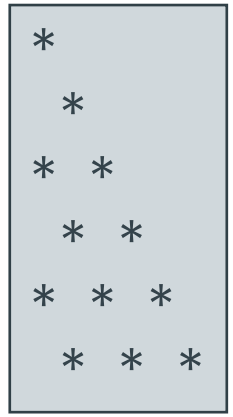


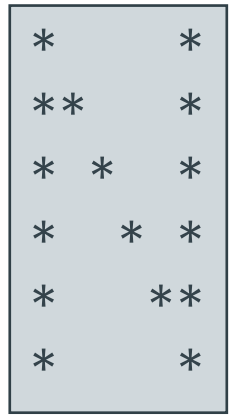


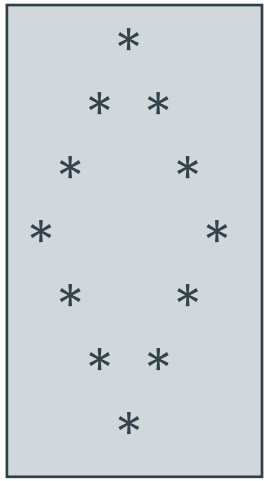


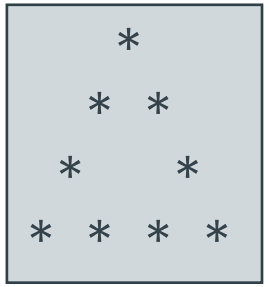












References

- Python Slides (2017) **– Department of Computer Engineering Kasetsart University**

- Think Python **– Allen B. Downey**