# CS Bridge, Lecture 5

## Control Flow Revisited


KOÇ UNIVERSITY

İZMİR DEMOKRASİ ÜNİVERSİTESİ 2016

LELAND STANFORD JUNIOR UNIVERSITY 1891

# Learning Goals

1. Constant Variables
2. If/elif/else statements
3. Comparison Operators
4. Random Library

How should we store information if it is known and never changes?

*Constants!*

# Constants

Constants are like variables that don't change

- Constants give descriptive names to literals

*Style note*

**constants**
Use constants with descriptive names instead of literals directly in your code.

# Constants

Constants are like variables that don't change

- Constants give descriptive names to literals

- Use all capital letters and snake_case when naming constants

*Style note*

**constant names**
Use all capital letters and snake_case, for example `MY_CONSTANT = 500`

# Constants

Constants are like variables that don't change

- Constants give descriptive names to literals
- Use all capital letters and snake_case when naming constants
- Constants are usually assigned outside functions and at the top of your program file (underneath the imports)

# Example of Using Constants

```
"""
File: constants.py
------------------An example program with constants
"""

INCHES_IN_FOOT = 12

def main():
    feet = float(input("Enter number of feet: "))
    inches = feet * INCHES_IN_FOOT
    print("That is " + str(inches) + " inches!")

# This provided line is required at the end of a
Python file
# to call the main() function.
if __name__ == '__main__':
    main()
```

# If/Else Revisited

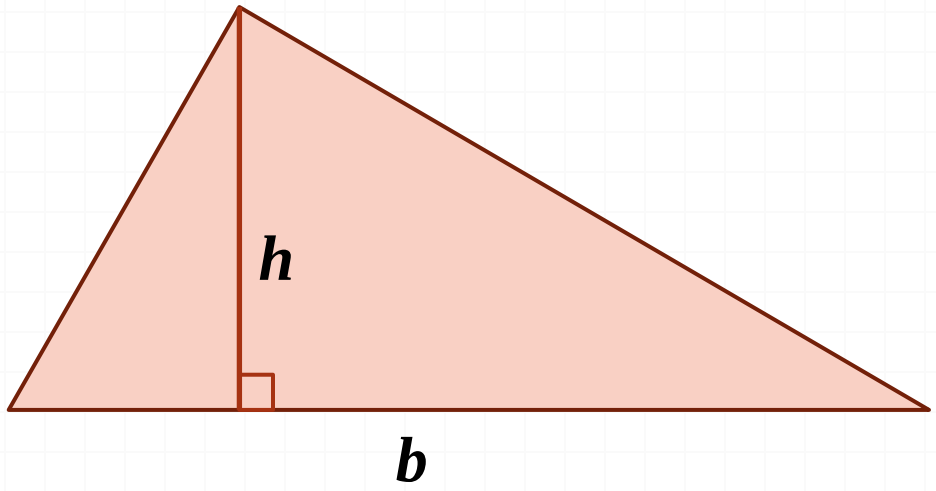
```
num = int(input("Enter a number: "))
if num == 0:
   print("Your number is 0 ")
else:
   if num > 0:
      print("Your number is positive")
   else:
      print("Your number is negative")
```

# Program-0

Area of a triangle

# Area of a triangle

- What is the area of this triangle?




$$Area = \frac{bh}{2}$$

# Area of a triangle

```
b= float(input('Enter base length:  '))
h= float(input('Enter height:  '))
print('')
area= b*h/2
print('Area of triangle with b=',b,'and h=',h,'is',area)
```

**Result:**

```
Enter base length:  8

Enter height:  4.5

Area of triangle with b= 8.0 and h= 4.5 is 18.0
```

# Area of a triangle

What if the user gives a negative value?

**Example:**

```
Enter base length:  4

Enter height:  -5

Area of triangle with b= 4.0 and h= -5.0 is -10.0
```

**Example:**

```
Enter base length:  -4

Enter height:  5

Area of triangle with b= -4.0 and h= 5.0 is -10.0
```

# Invalid values

- We cannot stop the user from giving negative (invalid) values; but we can detect them and choose not to do further evaluations with them.
- This requires writing our program with *branches* or *conditional statements* or *control flow*.
- In programming languages this is achieved with the **IF** command.
- The **IF** command involves a logical expression, which evaluates to a **TRUE** or a **FALSE**.

# Logical operators

- Logical operator **NOT** operates on one, **AND** and **OR** operate on two logical quantities.
- All three of them give a logical quantity (**TRUE** or **FALSE**) as a result.

| p | q | not p | p and q | p or q |
|---|---|---|---|---|
| **False** | **False** | True | False | False |
| **False** | **True** | True | False | True |
| **True** | **False** | False | False | True |
| **True** | **True** | False | True | True |

# Comparison Operators

- **Comparison operators** operate on (or compare) two comparable quantities of any type (integers, floats, strings, etc.)
- All of them give a logical quantity (**TRUE** or **FALSE**) as a result.

| Operator | Meaning |
|---|---|
| < | Is less than |
| > | Is greater than |
| <= | Is less than or equal to |
| >= | Is greater than or equal to |
| == | Is equal to |
| != | Is not equal to |

# = vs ==

In python:

**==**

is a **comparison operator**

**=**

is used for **variable assignment**

# Example

```
if 1 < 2 :
   print("1 is less than 2")
```

```
num = int(input("Enter a number: "))
if num == 0:
   print("That number is 0")
else :
   print("That number is not 0.")
```

# Opposite of logical expressions

- Assume we have a logical expression of the form:

$$p \otimes q$$

where $\otimes$ represents either **and** or **or** logical operator.

- The opposite of this expression is:

$$\text{not } (p \otimes q)$$

- which is:

$$(\text{not } p)\ (\text{not } \otimes)\ (\text{not } q)$$

# Opposite of logical expressions

- What are the opposites of the following expressions?

| | |
|---|---|
| a>2 | a<=2 |
| a==0 or b<5 | a!=0 and b>=5 |
| c>4 and is_even | c<=4 or not is_even |
| f==1 or f==2 or f==3 | f!=1 and f!=2 and f!=3 |

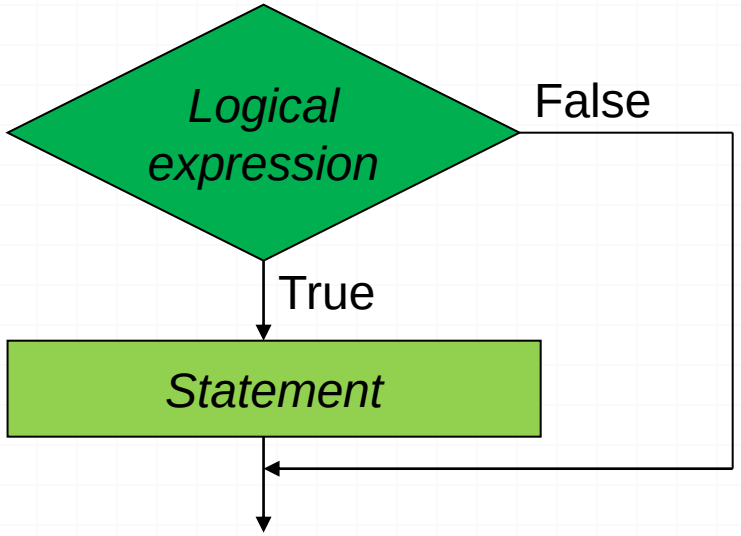
# Operator Precedence

- Parentheses ( ( ) )
- Power (**)
- Unary plus (+), unary minus (-)
- Multiplication (*), division (/), integer division(//), modulus (%)
- Addition (+), subtraction (-)
- Comparison operators (<, <=, >, >=, ==, !=)
- Logical NOT (not)
- Logical AND (and)
- Logical OR (or)

# Conditional statements

- The biggest power of computer programs come from their ability to do computations at a very fast rate.
- Their second most important property is the ability of making decisions (by use of conditional statements).
- The main building block of a conditional statement is a logical expression that yields a TRUE or FALSE value.
- We will now explore different ways of building conditional statements (or *control flow*).

# IF statement (1)



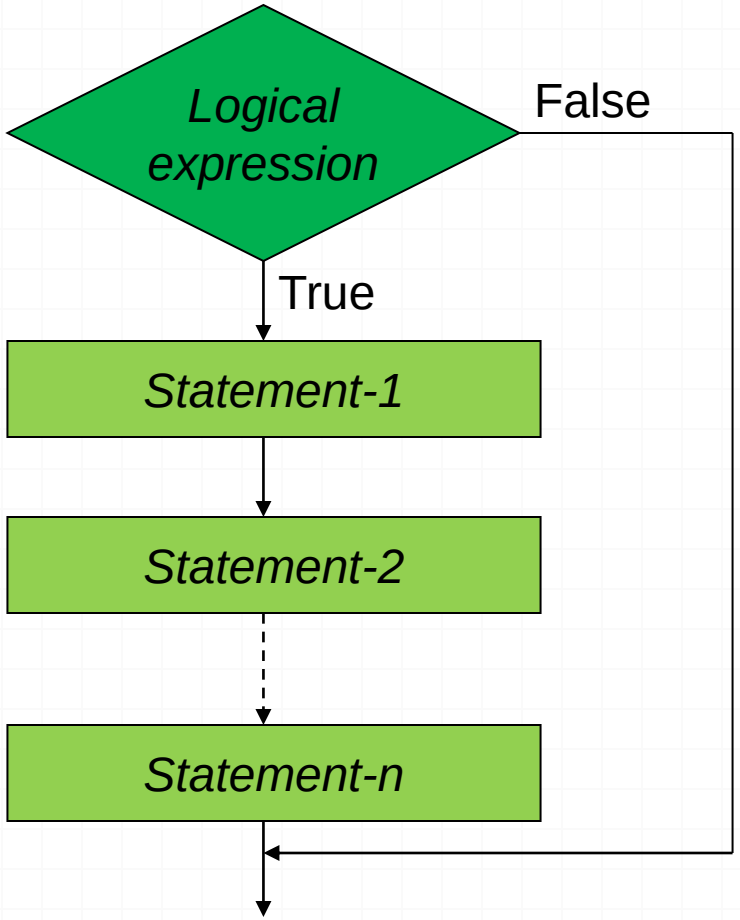

```
if logical-expression:
    statement
```

# IF statement (1) example

```
grade= int(input('Enter your exam grade: '))

if grade>=90:
    print('Well done!')
```

# IF statement (2)

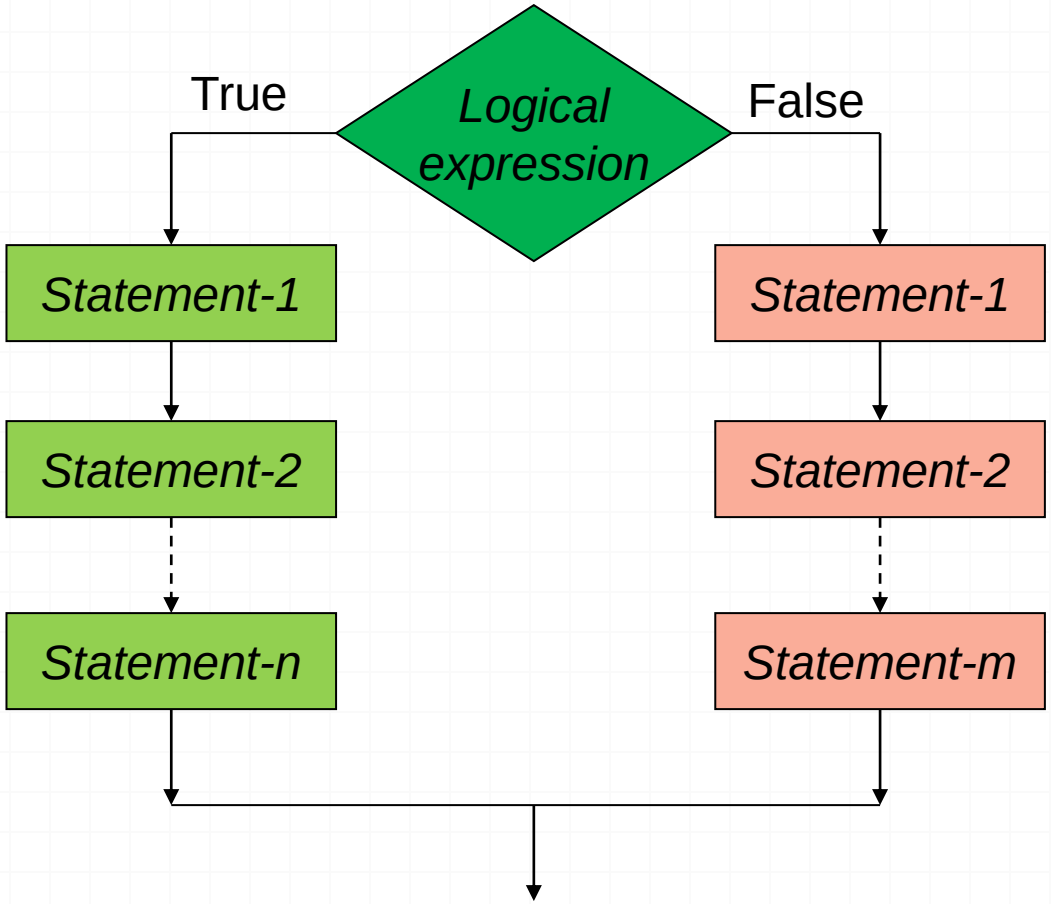



```
if logical-expression:
    statement-1
    statement-2
    ...
    statement-n
```

# IF statement (2) example

```
grade= int(input('Enter your exam grade: '))
if grade>=90:
    print('Well done!')
    print('You are an A student')
```

# IF statement (3)

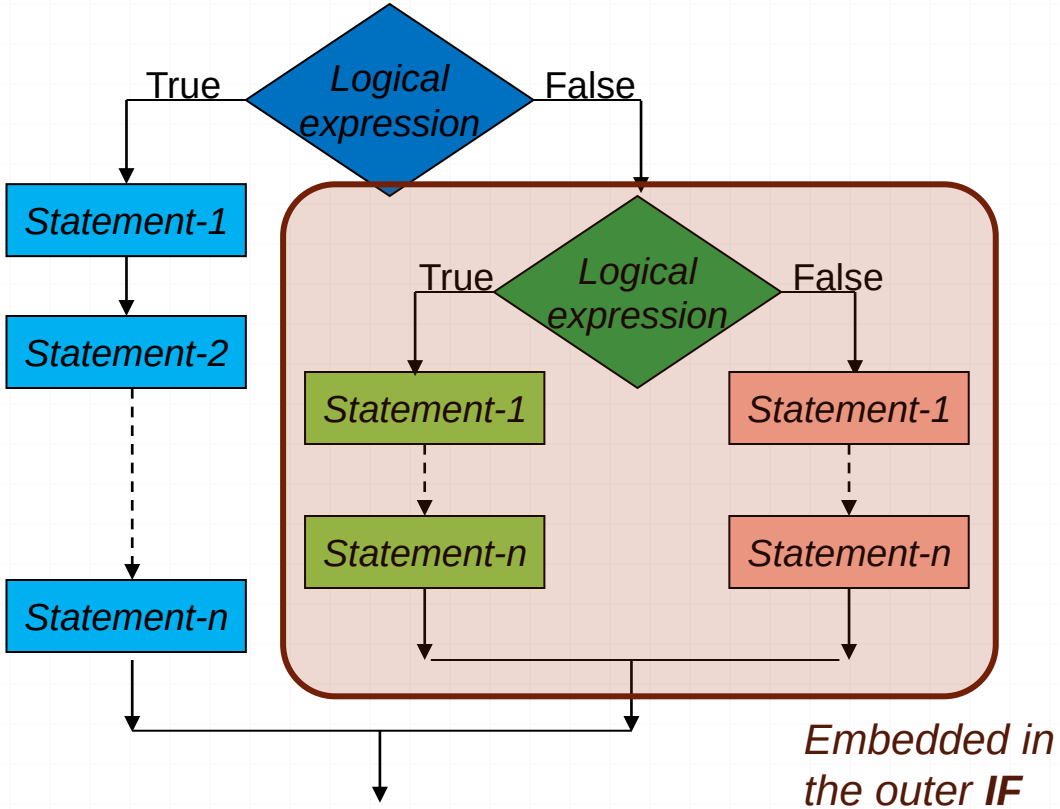



```
if  logical-expression:
        statement-1
        ...
        statement-n
else:
        statement-1
        ...
        statement-m
```

# IF statement (3) example

```
grade= int(input('Enter your exam grade: '))
if grade>=50:
    print('You pass.')
else:
   print('You fail.')
   print('Try harder next time.')
```

# IF statement (4)

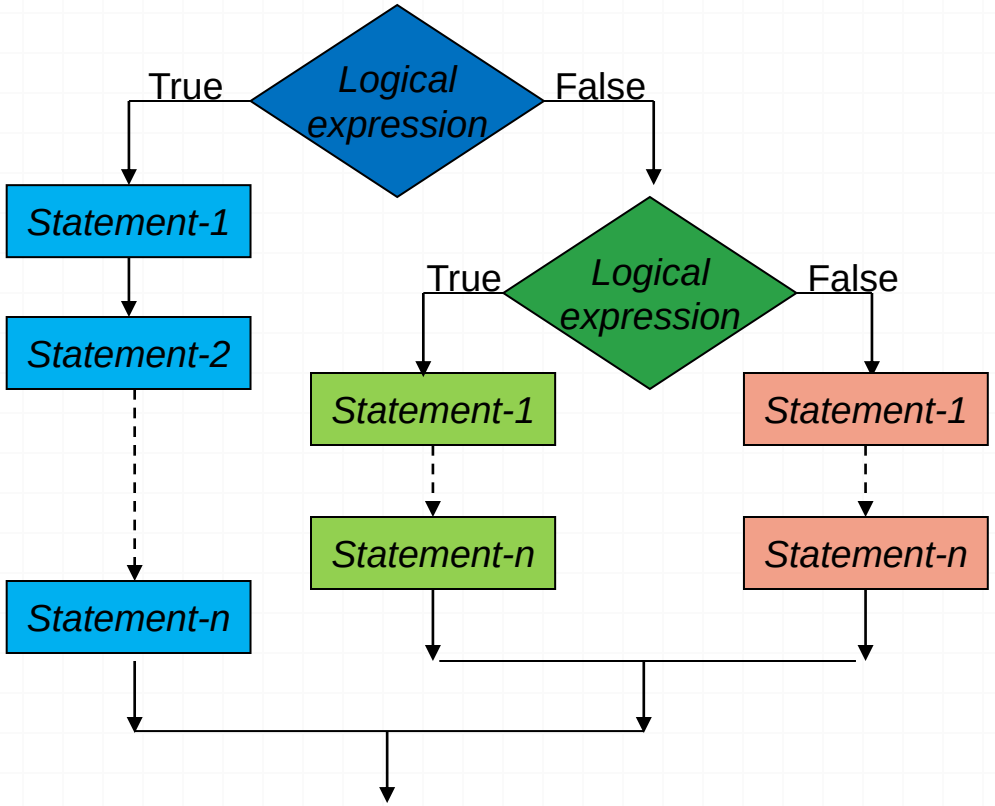



```
if  logical-expression:
    statement-1
    statement-2
    ...
    statement-n
else:
    if logical-expression:
        statement-1
        ...
        statement-n
    else:
        statement-1
        ...
        statement-n
```

# IF statement (4) example

```
age= int(input('Enter your age: '))
if age<13:
    print('You are a child.')
else:
    if age>=18:
        print('You are an adult.')
    else:
        print('You are a teenager.')
```

# IF statement (5)




```
if logical-expression:
    statement-1
    statement-2
    ...
    statement-n
elif logical-expression:
    statement-1
    ...
    statement-n
else:
    statement-1
    ...
    statement-n
```

# IF statement (5) example

```
age= int(input('Enter your age: '))
if age<13:
    print('You are a child.')
elif age>=18:
    print('You are an adult.')
else:
    print('You are a teenager.')
```

# IF command general format

```
if <logical expression-1>:
    <some commands>
elif <logical expression-2>:
    <some commands>
elif <logical expression-3>:
    <some commands>
elif <logical expression-4>:
    <some commands>
…
else:
    <some commands>
```

You can place any number of **ELIF** parts into an **IF** block.

The **ELSE** part, if it exists, is always the last branch.

Note that there is always a logical expression on an **ELIF** line. There is never a logical expression on the **ELSE** line.

**ELSE** can be interpreted as "if all previous logical expression tests have failed, then do this".

# Program-1

Area of a triangle improved

# Area of a triangle

We were discussing the following problem.

**Example:**

```
Enter base length:   4

Enter height:   -5

Area of triangle with b= 4.0 and h= -5.0 is -10.0
```

**Example:**

```
Enter base length:   -4

Enter height:   5

Area of triangle with b= -4.0 and h= 5.0 is -10.0
```

# Area of a triangle

This is what we want:

**Example:**

```
Enter base length:   -4

Enter height:   5

Base must be positive!
```

**Example:**

```
Enter base length:   4

Enter height:   -5

Height must be positive!
```

Fix it with an IF structure

# Area of a triangle

```
# Version 1 with ELSE and embedded IF
b= float(input('Enter base length:  '))
h= float(input('Enter height:  '))
print('')
if b<=0:
    print('Base must be positive!')
else:
    if h<=0:
        print('Height must be positive!')
    else:
        area= b*h/2
        print('Area of triangle with b=',b,'and h=',h,'is',area)
```

# Area of a triangle

```
# Version 2 with ELIF
b= float(input('Enter base length:  '))
h= float(input('Enter height:  '))
print('')
if b<=0:
    print('Base must be positive!')
elif h<=0:
    print('Height must be positive!')
else:
    area= b*h/2
    print('Area of triangle with b=',b,'and h=',h,'is',area)
```

# Area of a triangle

What happens when both base and height are negative?

**Example:**

```
Enter base length:  -4

Enter height:  -5

Base must be positive!
```

Fix it with a better IF structure

# Area of a triangle

```
# Version 3 handles both b and h being negative
b= float(input('Enter base length:  '))
h= float(input('Enter height:  '))
print('')
if b<=0 and h<=0:
    print('Both base and height must be positive!')
elif b<=0:
    print('Base must be positive!')
elif h<=0:
    print('Height must be positive!')
else:
    area= b*h/2
    print('Area of triangle with b=',b,'and h=',h,'is',area)
```

# Area of a triangle

```
# Version 4 with independent IFs
# Prints two warning lines if both negative
b= float(input('Enter base length:  '))
h= float(input('Enter height:  '))
print('')
if b<=0:
    print('Base must be positive!')
if h<=0:
    print('Height must be positive!')
if b>0 and h>0:
    area= b*h/2
    print('Area of triangle with b=',b,'and h=',h,'is',area)
```

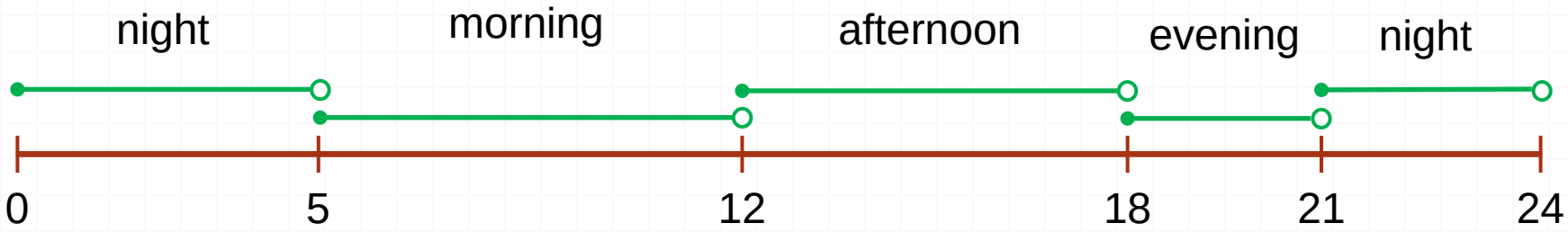
# Area of a triangle

```
# Version 5 prints one generic message
# in case of any invalid input
b= float(input('Enter base length:  '))
h= float(input('Enter height:  '))
print('')
if b<=0 or h<=0:
    print('Base and height must be positive!')
else:
    area= b*h/2
    print('Area of triangle with b=',b,'and h=',h,'is',area)
```

# Program-2

Day of time greeting

# Day of time greeting




**Example:**

```
What hour is it?  2
Good night
```

**Example:**

```
What hour is it?  18
Good evening
```

Write your program

# Day of time greeting

```
h= int(input('What hour is it?  '))
if h>=0 and h<5 or h>=21 and h<24:
    print('Good night')
elif h>=5 and h<12:
    print('Good morning')
elif h>=12 and h<18:
    print('Good afternoon')
elif h>=18 and h<21:
    print('Good evening')
else:
    print('Are you living on Mars???')
```

# Day of time greeting

```
h= int(input('What hour is it?  '))
if h<0 or h>=24:
    print('Are you living on Mars???')
elif h<5 or h>=21:
    print('Good night')
elif h<12:
    print('Good morning')
elif h<18:
    print('Good afternoon')
else:
    print('Good evening')
```

# Random numbers

# Guess My Number

GuessMyNumber

```
I am thinking of a number between 0 and 99...
Enter a guess: 50
Your guess is too high

Enter a new number: 25
Your guess is too low

Enter a new number: 40
Your guess is too low

Enter a new number: 45
Your guess is too low

Enter a new number: 48
Congrats! The number was: 48
```

# Random numbers

- Python has a built-in module for generating random numbers.
- You have to include the following statement at the beginning of your program:

```
import random
```

- There are only a few functions we will use from this module.

# Random numbers

- **randint( )** is a function for generating a random integer.
- It requires a first value and a last value as argument:

```
random.randint(first,last)
```

- The result is any number between `[first,last]` (both inclusive).
- Example:

```
for i in range(10):
    print(random.randint(1,6),end=' ')
```

- Result:

```
6 2 1 5 5 2 3 1 6 2
```

# Random numbers

- **random( )** is a function for generating a random floating point number.
- It requires no arguments:

```
random.random()
```

- The result is any number between 0.0 (inclusive) and 1.0 (exclusive).
- Example:

```
a= random.random()
```

- Result:

```
0.7285270343303428
```

# Guess My Number

```
secret_number = random.randint(1, 99)
print("I am thinking of a number between 1 and 99...")
guess = int(input("Enter a guess: "))
# True if guess is not equal to secret number
while guess != secret_number:
        # True if guess is less than secret number
        if guess < secret_number:
                print("Your guess is too low")
        else:
                print("Your guess is too high")

        print("") # an empty line
        guess = int(input("Enter a new guess: "))

print("Congrats! The number was: " , secret_number)
```

# The end