SCIENTIFIC COMPUTING WITH PYTHON

Operators

Course Coordinator: Dr. R. Mariyal Jebasty
Assistant Professor,
Department of Physics
Wavoo Wajeeha College of Arts & Science
Kayalpatnam.

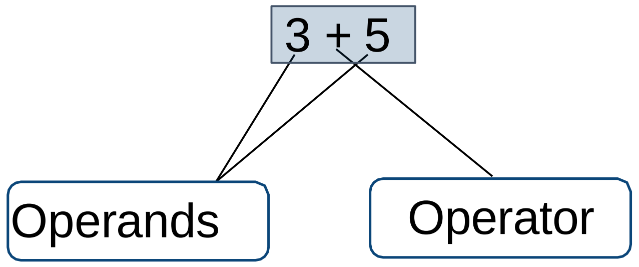
Course Instructors

1. Mrs. Pushpa, Assistant Professor in Physics, Wavoo Wajeeha Women's College of Arts & Science, Kayalpatnam.
2. Dr. S. Usharani , Assistant Professor in Physics, Wavoo Wajeeha Women's College of Arts & Science, Kayalpatnam.

Operators

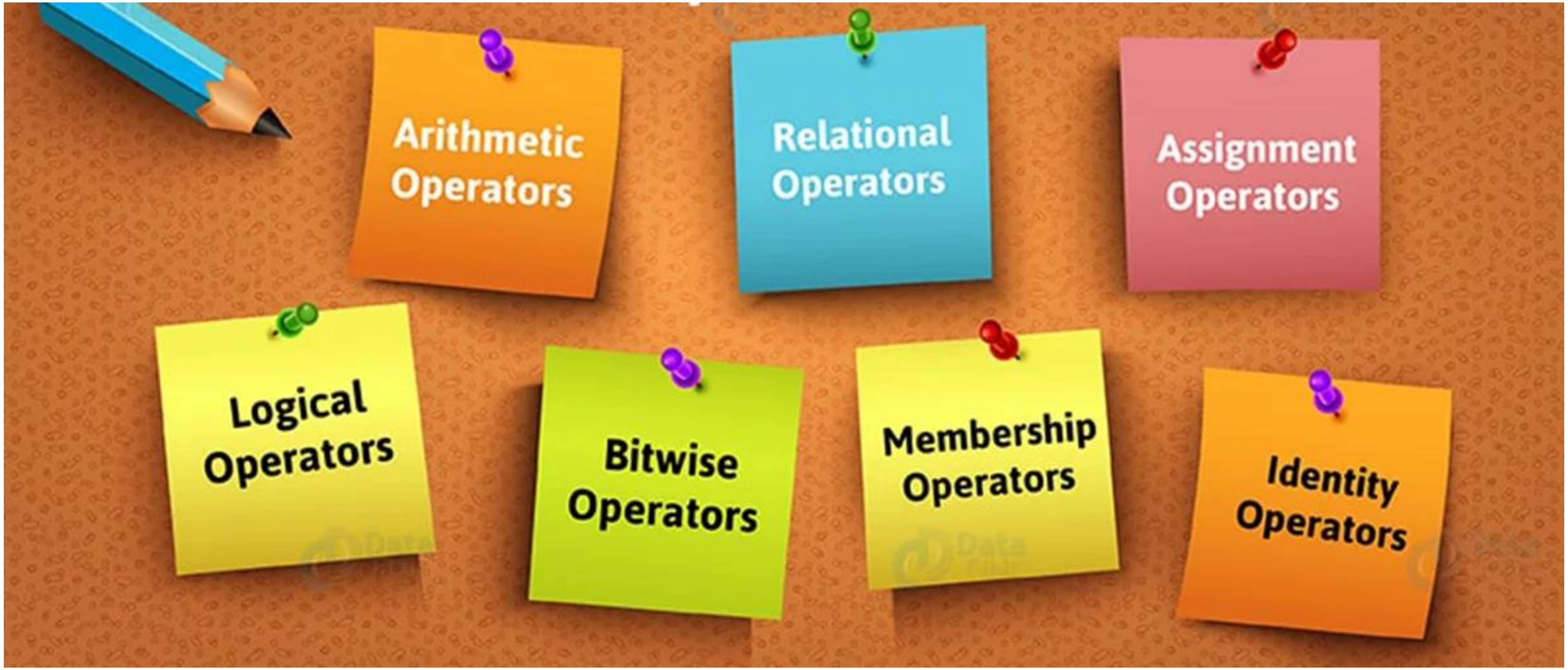
- What are Operators?

- Symbols that represent mathematical or logical tasks.



Task	Symbol
Addition	+
Subtraction	-
Multiplication	*
Division	/

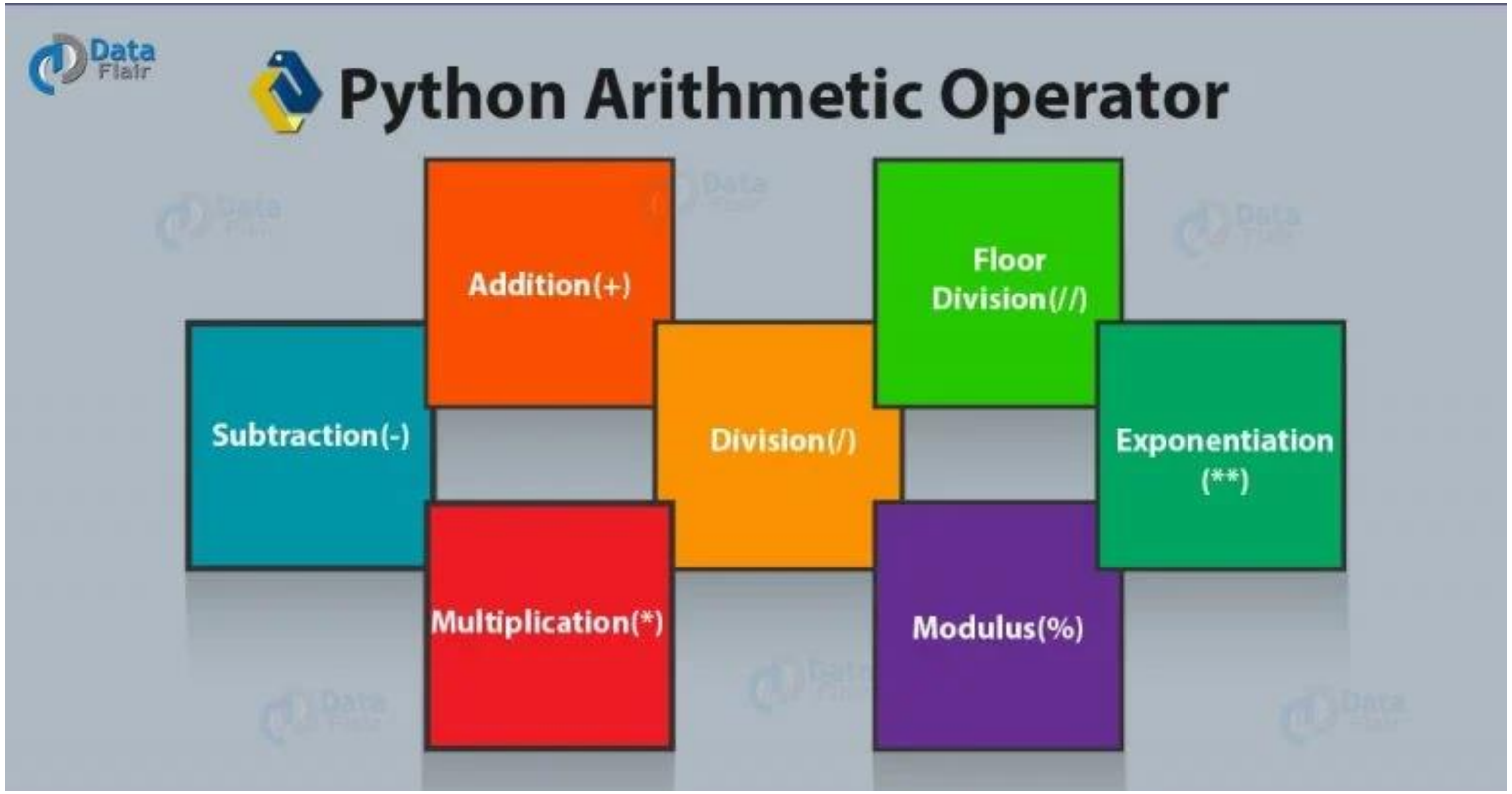
Different Types of Operators



Different Types of Operators

Arithmetic Operators

- Addition +
- Subtraction –
- Multiplication *
- Division /
- Modulo %
- Floor division //
- Exponent **



Different Types of Operators

Arithmetic Operators

- Addition +
- Subtraction –
- Multiplication *
- Division /
- Modulo %
- Floor division //
- Exponent **

Addition -> $\boxed{3 + 5 + 6} = 14$

Subtraction -> $\boxed{6 - 4 - 2} = 0$

Different Types of Operators

Arithmetic Operators

- Addition +
- Subtraction –
- Multiplication *
- Division /
- Modulo %
- Floor division //
- Exponent **

Addition ->	3 + 5 + 6	= 14
Subtraction ->	6 - 4 - 2	= 0
Multiplication ->	3 * 5 * 6	= 90
Division ->	5 / 3	= 1.67

Different Types of Operators

Arithmetic Operators

- Addition +
- Subtraction –
- Multiplication *
- Division /
- Modulo %
- Floor division //
- Exponent **

Addition	->	3 + 5 + 6	= 14
Subtraction	->	6 - 4 - 2	= 0
Multiplication	->	3 * 5 * 6	= 90
Division	->	5 / 3	= 1.67
Modulo	->	5 % 3	= 2
Floor Division	->	5 // 3	= 1

Different Types of Operators

Arithmetic Operators

- Addition +
- Subtraction –
- Multiplication *
- Division /
- Modulo %
- Floor division //
- Exponent **

Addition ->	3 + 5 + 6	= 14
Subtraction ->	6 - 4 - 2	= 0
Multiplication ->	3 * 5 * 6	= 90
Division ->	5 / 3	= 1.67
Modulo ->	5 % 3	= 2
Floor Division ->	5 // 3	= 1
Exponent ->	5 ** 3	= 125

Different Types of Operators

Relational Operators

- Less than <
- Less than or Equal to <=
- Equal to ==
- Greater than >
- Greater than or equal to >=

Not equal to !=

Different Types of Operators

Relational Operators

- Less than <
- Less than or Equal to <=
- Equal to ==
- Greater than >
- Greater than or equal to >=

Not equal to !=

Less than -> x < y -> ?

x	y
3	5

Different Types of Operators

Relational Operators

- Less than <
- Less than or Equal to <=
- Equal to ==
- Greater than >
- Greater than or equal to >=

Not equal to !=

Less than -> x < y -> True!

x $\boxed{3}$ y $\boxed{5}$

Different Types of Operators

Relational Operators

- Less than <
- Less than or Equal to <=
- Equal to ==
- Greater than >
- Greater than or equal to >=

Not equal to !=

Less than -> x < y -> ?

x	y
17	5

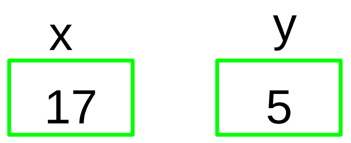
Different Types of Operators

Relational Operators

- Less than <
- Less than or Equal to <=
- Equal to ==
- Greater than >
- Greater than or equal to >=

Not equal to !=

Less than -> $\boxed{x < y}$ -> False



Different Types of Operators

Relational Operators

- Less than <
- Less than or Equal to <=
- Equal to ==
- Greater than >
- Greater than or equal to >=

Not equal to !=

x	y
17	5

Less than -> x < y

Less than or equal to -> x <= y

Equal to -> x == y

Different Types of Operators

Relational Operators

- Less than <
- Less than or Equal to <=
- Equal to ==
- Greater than >
- Greater than or equal to >=
- Not equal to !=

x	y
17	5

Less than -> x < y

Less than or equal to -> x <= y

Equal to -> x == y

Greater than -> x > y

Greater than or equal to -> x <= y

Not Equal to -> x != y

Different Types of Operators

Logical Operators

- and
- or
- not

Different Types of Operators

Logical Operators

- and
- or
- not

and -> True only if both comparisons are True.

Different Types of Operators

Logical Operators

- and
- or
- not

and -> True only if both comparisons are True.
x < 5 and y > 8 --> ?

x	y
3	7

Different Types of Operators

Logical Operators

- and
- or
- not

and -> True only if both comparisons are True.
x < 5 and y > 8 --> True and False

x	y
3	7

Different Types of Operators

Logical Operators

- and
- or
- not

x	y
3	7

and -> True only if both comparisons are True.
x < 5 and y > 8 --> True and False --> False

Different Types of Operators

x
3

y
7

Logical Operators

- and
- or
- not

and -> True only if both comparisons are True.

x < 5 and y > 8 --> True and False --> False

or -> True if either of the comparisons are True.

x < 5 or y > 8 --> True or False

Different Types of Operators

Logical Operators

- and
- or
- not

x	y
3	7

and -> True only if both comparisons are True.

x < 5 and y > 8 --> True and False --> False

or -> True if either of the comparisons are True.

x < 5 or y > 8 --> True or False --> True

Different Types of Operators

x: 3   y: 7

Logical Operators

- and
- or
- not

and -> True only if both comparisons are True.

x < 5 and y > 8 --> True and False --> False

or -> True if either of the comparisons are True.

x < 5 or y > 8 --> True or False --> True

not -> True if comparison is False and vice-versa.

not x < 5 --> not True

Different Types of Operators

x: 3   y: 7

Logical Operators

- and
- or
- not

and -> True only if both comparisons are True.

x < 5 and y > 8 --> True and False --> False

or -> True if either of the comparisons are True.

x < 5 or y > 8 --> True or False --> True

not -> True if comparison is False and vice-versa.

not x < 5 --> not True --> False

Summary

Arithmetic Operators

- Addition +
- Subtraction –
- Multiplication *
- Division /
- Modulo %
- Floor division /
- Exponent **

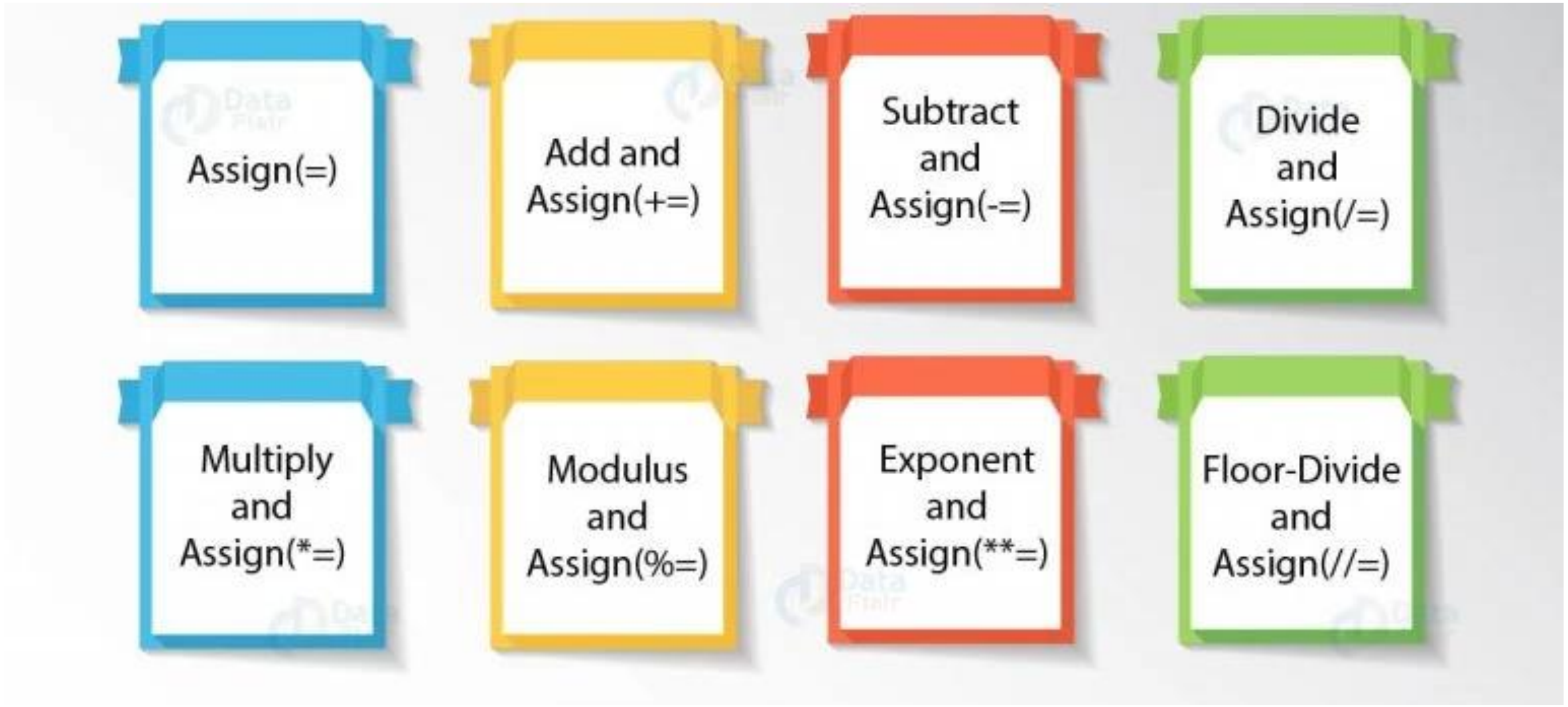
Relational Operators

- Less than <
- Less than or Equal to <=
- Equal to ==
- Greater than >
- Greater than or equal to >=
- Not equal to !=

Logical Operators

- and
- or
- not

Python Assignment Operator



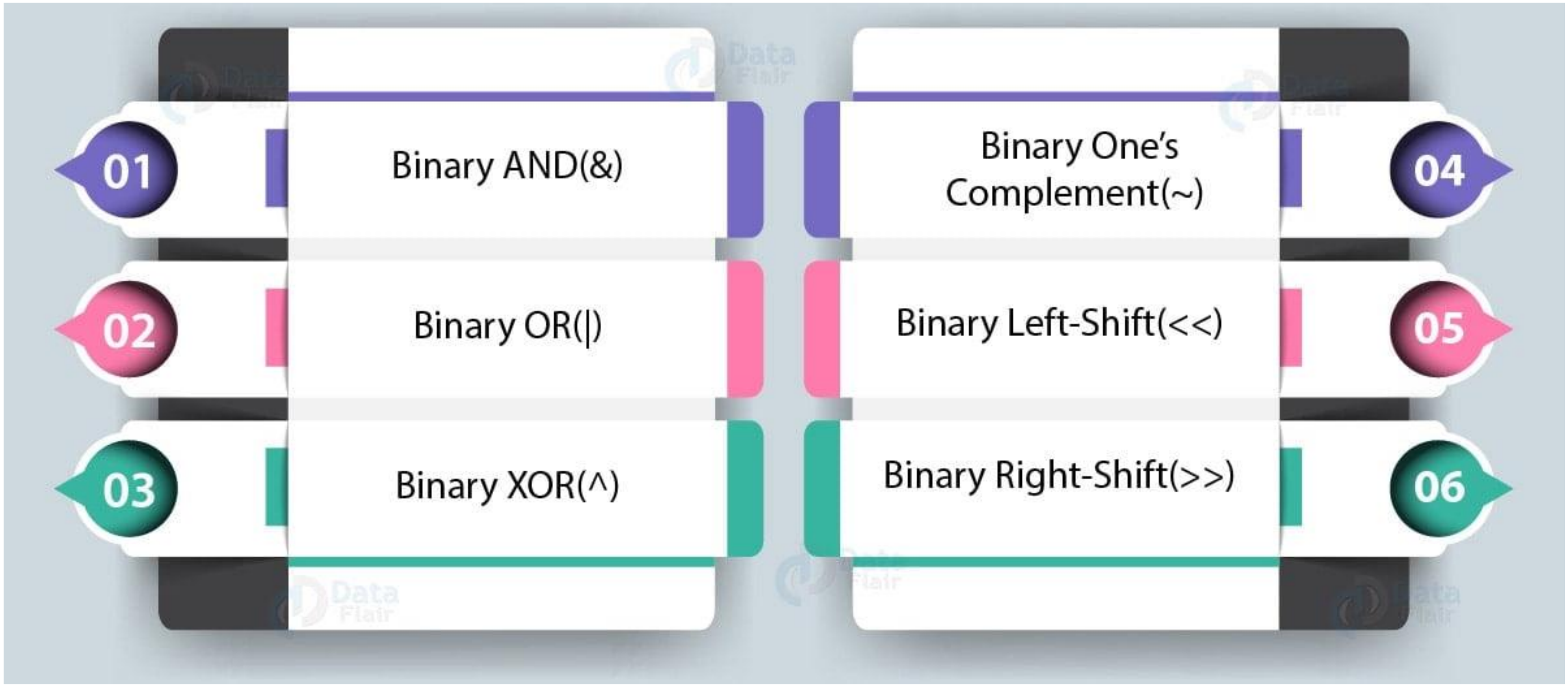
Python Identity Operator

- is Operator in Python
- is not Operator in Python

Python Membership Operator

- operators test whether a value is a member of a **sequence**. The sequence may be a **list**, a **string**, or a **tuple**.
- We have two membership python operators- ‘in’ and ‘not in’.

Python Bitwise Operator



Example

- **i/p:** 1 & 3
- **o/p:** 2
- It perform Bit by bit AND operation
- Here, binary for 1 is 01, and that for 3 is 11. &-ing them results in 01, which is binary for 1.

Looping Statements

Scenario

- Print “Python is awesome” 1000 times

```
print(“Python is awesome”)
print(“Python is awesome”)
print(“Python is awesome”)
print(“Python is awesome”)
print(“Python is awesome”)
print(“Python is awesome”)
print(“Python is awesome”)
```

```
print(“Python is awesome”)
print(“Python is awesome”)
…….
…….
…….
…….
…….
```

Scenario

- Print “Python is awesome” 1000 times

```
print(“Python  is  awesome”)
print(“Python  is  awesome”)
print(“Python  is  awesome”)
print(“Python  is  awesome”)
print(“Python  is  awesome”)
print(“Python  is  awesome”)
print(“Python is awesome”)
```

Looping Statements!!

```
print(“Python is awesome”)
print(“Python is awesome”)
…….
…….
…….
…….
…….
```

Looping constructs in Python

- Print “Python is awesome” 1000 times

Pseudo-Code:

Looping constructs in Python

- Print “Python is awesome” 1000 times

Pseudo-Code:

```
print “Python is awesome” (repeat 1000 times)
```

Looping constructs in Python

- Print “Python is awesome” 1000 times

Pseudo-Code:

```
print “Python is awesome” (repeat 1000 times)
```

Code:

```
for i in range(1000):
   print(“Python is awesome”)
```

Looping constructs in Python: The for loop

- **The 'for' loop in python**

Pseudo-Code:

```
print “Python is awesome” (repeat 1000 times)
```

Syntax:

```
for iterating_variable in sequence:
   statements(s)
```

Looping constructs in Python: The for loop

- **The 'for' loop in python**

Pseudo-Code:

```
print "Python is awesome" (repeat 1000 times)
```

Syntax:

```
for iterating_variable in sequence:
    statements(s)
```

4 space "indentation"

Looping constructs in Python: The for loop

- **The ‘for’ loop in python**

Pseudo-Code:

```
print “Python is awesome” (repeat 1000 times)
```

Code:

```
for i in range(1000):
    print(“Python is awesome”)
```

Looping constructs in Python: The for loop

- **The ‘for’ loop in python**

Pseudo-Code:

```
print “Python is awesome” (repeat 1000 times)
```

Code

```
for i in range(1000):
    print(“Python is awesome”)
```

4 space “indentation”

Looping constructs in Python: The for loop

- **The 'for' loop in python**

Pseudo-Code:

```
print "Python is awesome" (repeat 1000 times)
```

Code

```
for i in range(1000):
    print("Python is awesome")
```

"stop condition"

4 space "indentation"

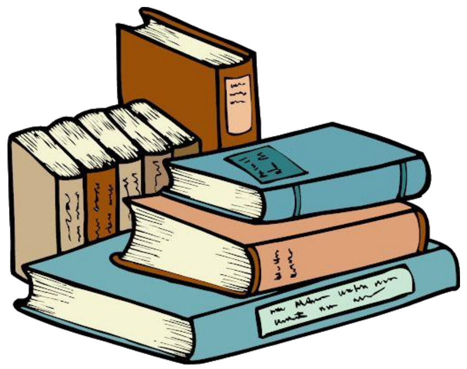
Types of loops: Example

Types of loops: Example

Scenario #1:

Pass in 5 subjects:

- Math
- Physics
- English..etc.

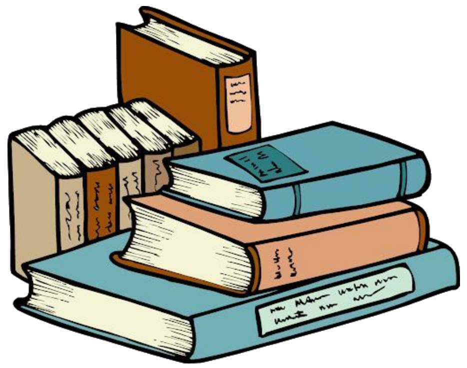


Types of loops: Example

Scenario #1:

Pass in 5 subjects:

- Math
- Physics
- English..etc.



Code:

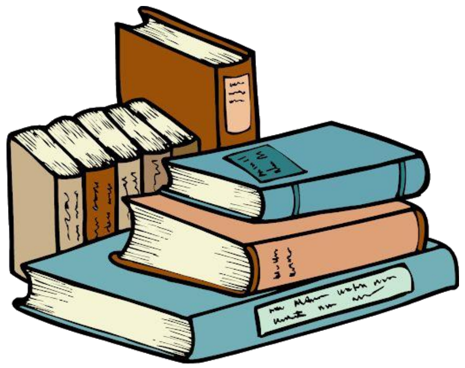
```
for subject in range(5):
    # Pass exam 5 times
    # Once for each subject
```

Types of loops: Example

Scenario #1:

Pass in 5 subjects:

- Math
- Physics
- English..etc.



“stop condition”

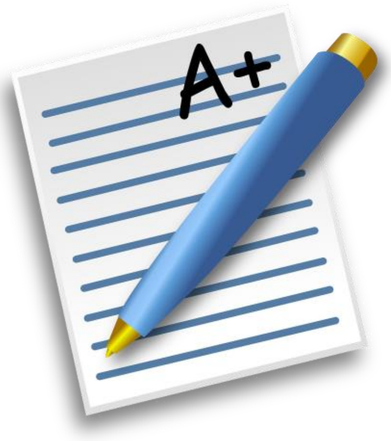
Code:

```
for subject in range(5):
    # Pass exam 5 times
    # Once for each subject
```

Types of loops: Example

Scenario #2:

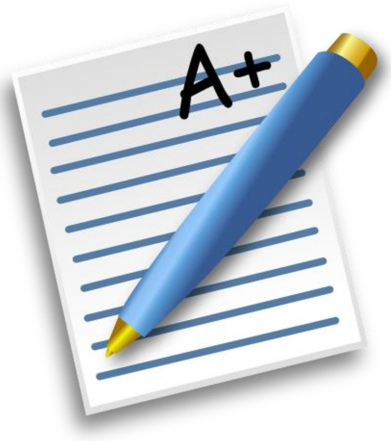
- Secure at least an **A grade** in Math to pass.



Types of loops: Example

Scenario #2:

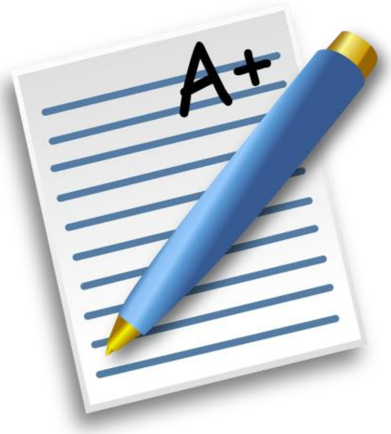
- Secure at least an **A grade** in Math to pass.
- When do you stop?



Types of loops: Example

Scenario #2:

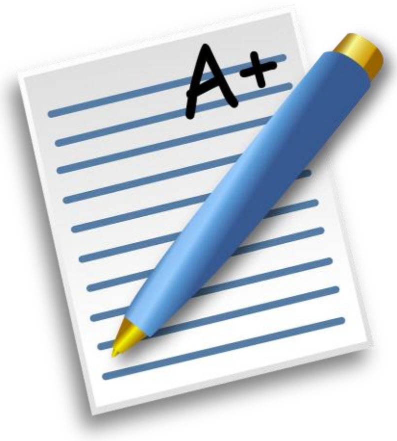
- Secure at least an **A grade** in Math to pass.
- When do you stop?
- Keep trying until you succeed!
- New kind of looping needed..



Types of loops: Example

Scenario #2:

- Secure at least an **A grade** in Math to pass.
- When do you stop?
- Keep trying until you succeed!
- New kind of looping needed..



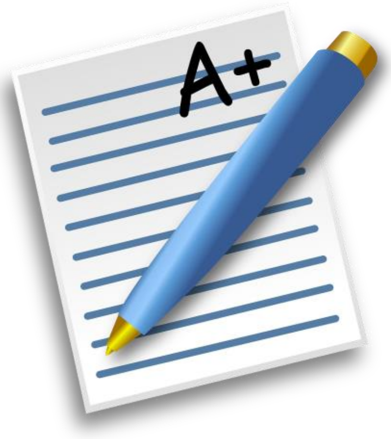
Code:

```
while grade != ‘A’:
    # Keep repeating until
    # comparison gives a False
```

Types of loops: Example

Scenario #2:

- Secure at least an **A grade** in Math to pass.
- When do you stop?
- Keep trying until you succeed!
- New kind of looping needed..



comparison (“stopping criteria”)

Code:

```
while grade != ‘A’:
    # Keep repeating until
    # comparison gives a False
```

Looping constructs in Python: The while loop

- **The ‘while’ loop in python**

Syntax:

```
while comparison:
   statements(s)
```

Code:

```
while grade != ‘A’:
    # Keep repeating until
    # comparison gives a False
```

Looping constructs in Python: Summary

‘for’ loop

Code:

```
for subject in range(5):
    # Pass exam 5 times
    # Once for each subject
```

‘while’ loop

Code:

```
while grade != ‘A’:
    # Keep repeating until
    # comparison gives a False
```

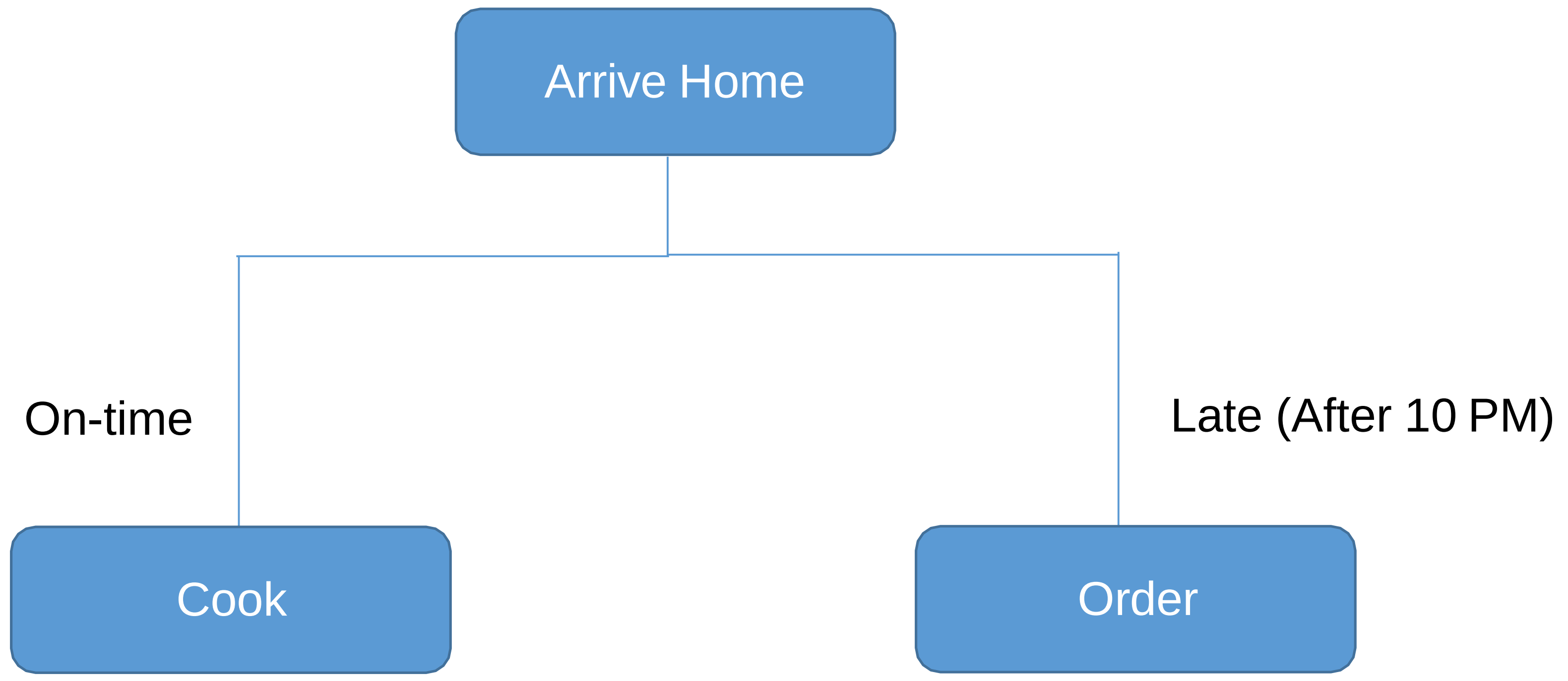
Conditional Statements

Conditional Statements: Scenario

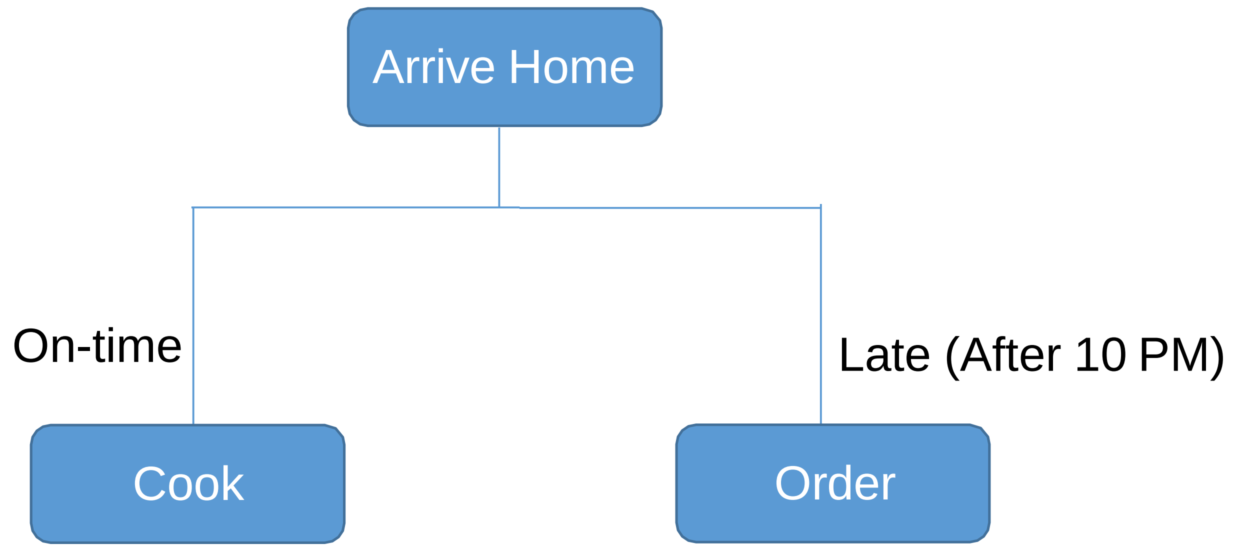
Example

Conditional Statements: Scenario

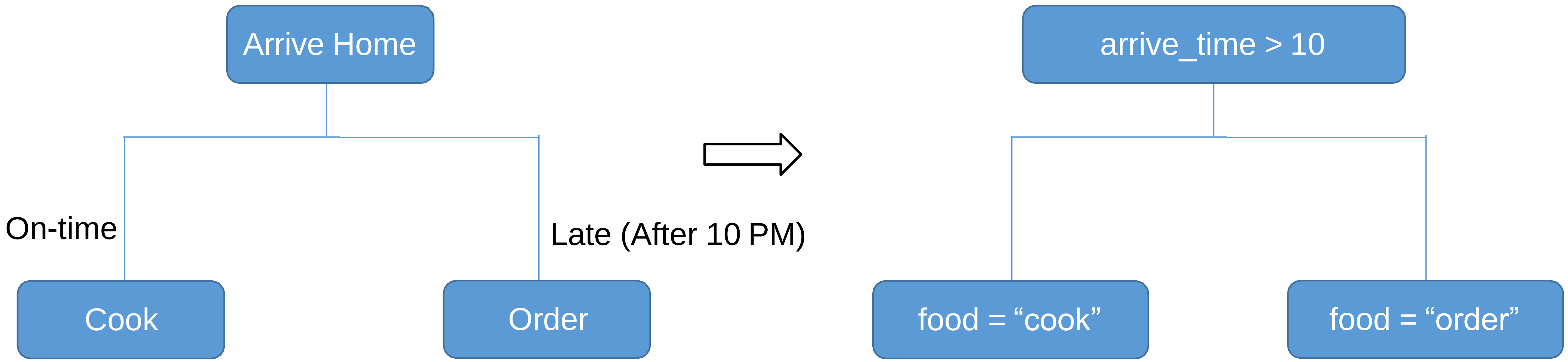
Example



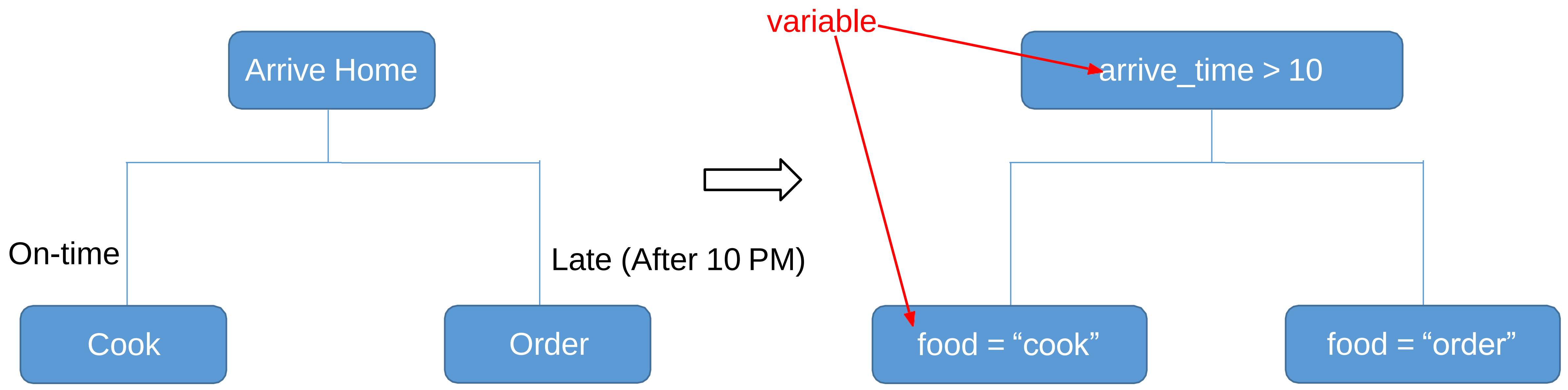
Conditional Statements



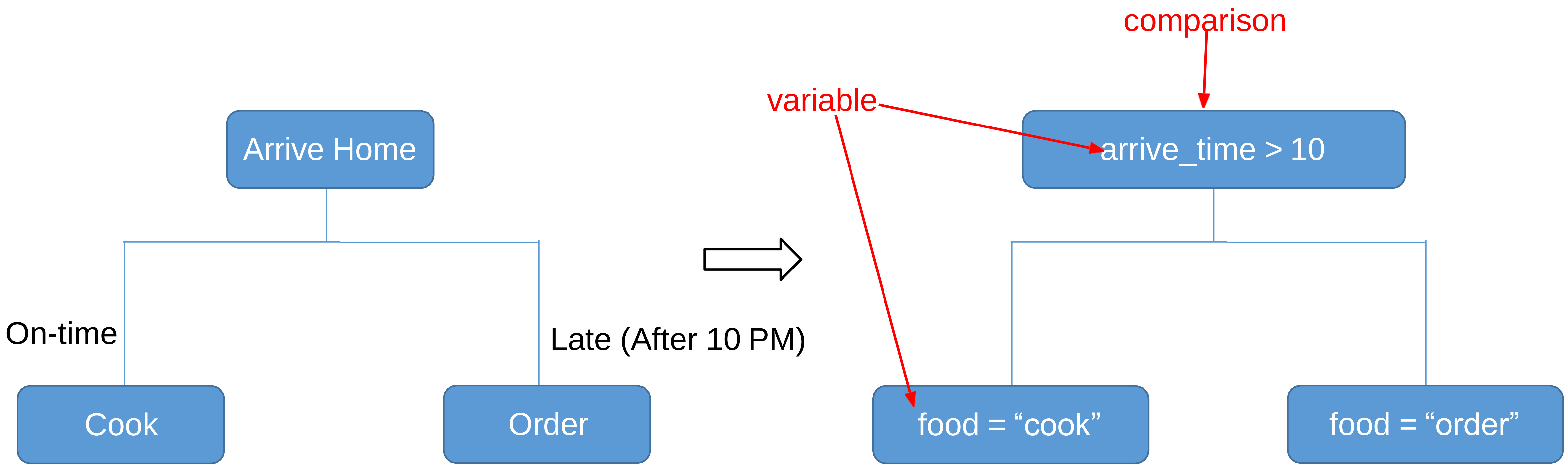
Conditional Statements



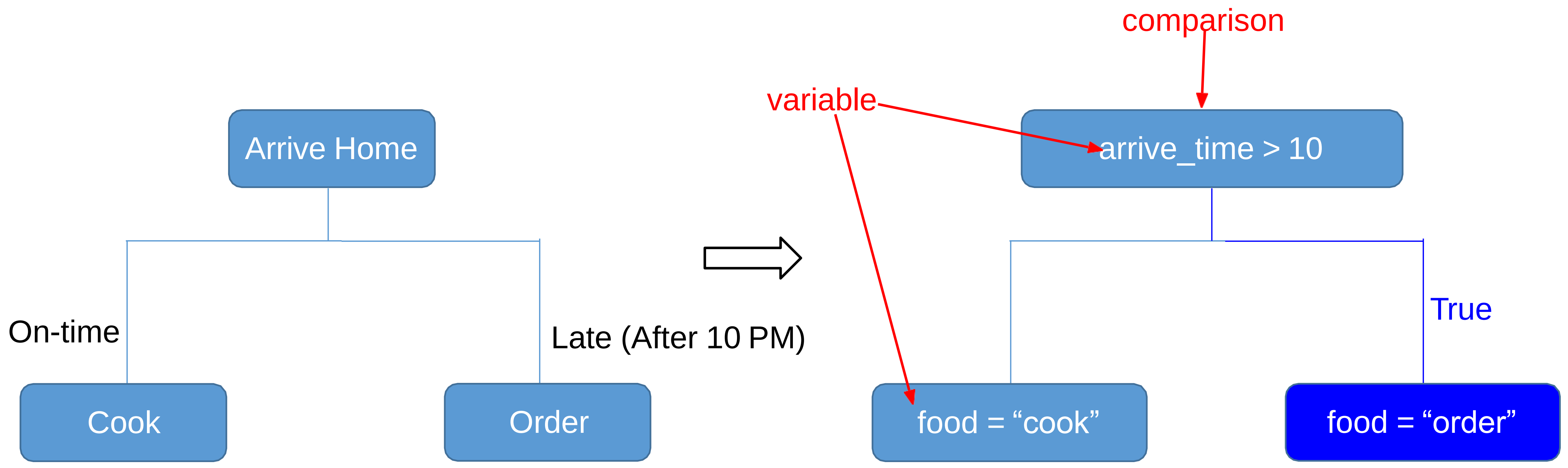
Conditional Statements



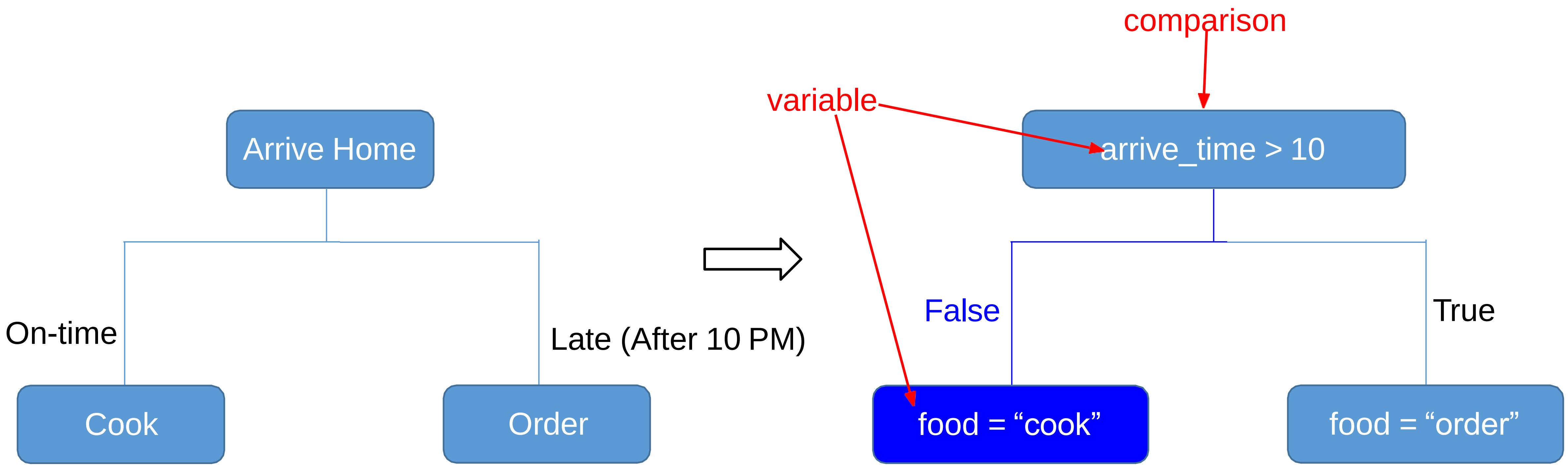
Conditional Statements



Conditional Statements

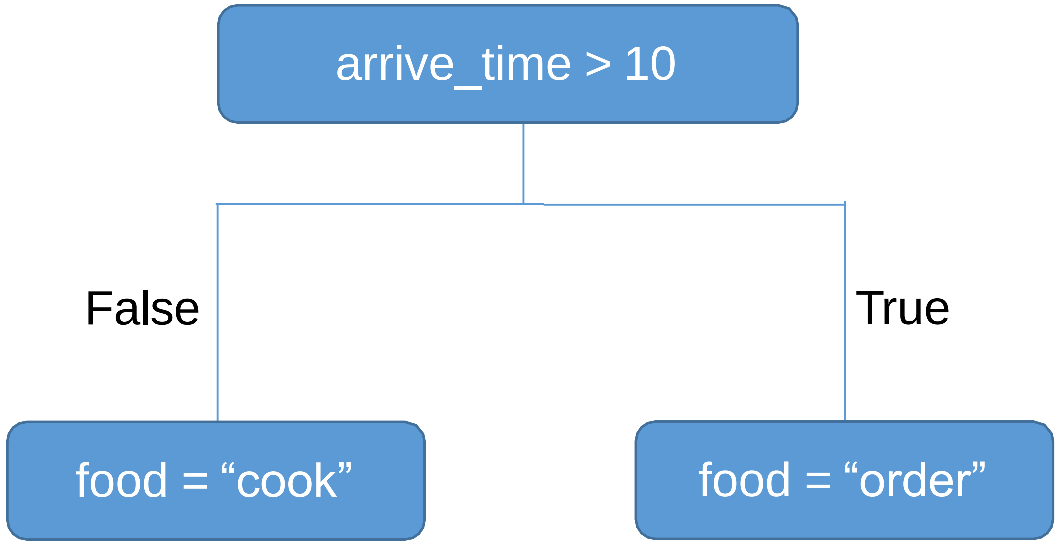


Conditional Statements



Conditional Statements

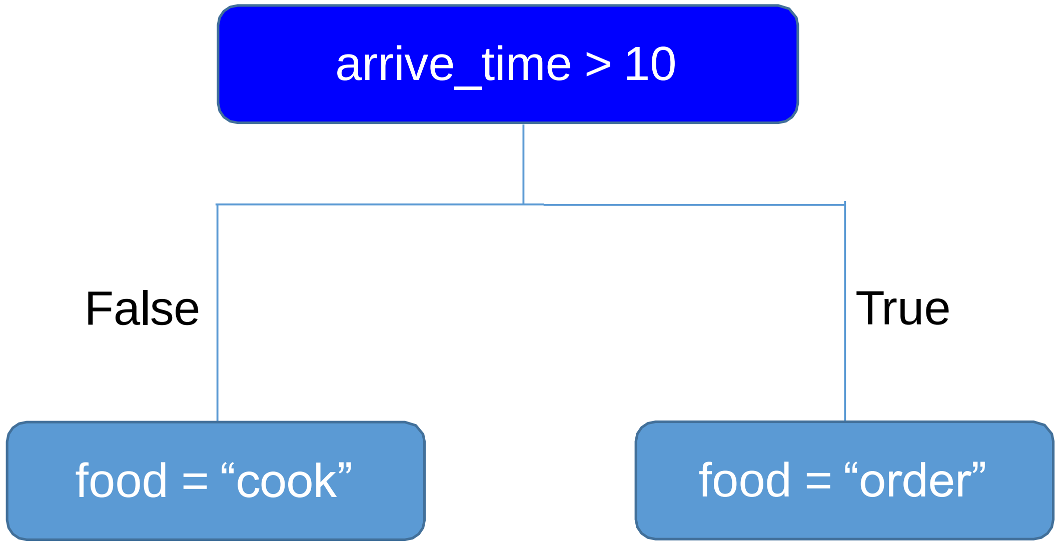
Pseudo-Code



Conditional Statements

Pseudo-Code

```
Check if arrive_time > 10
```

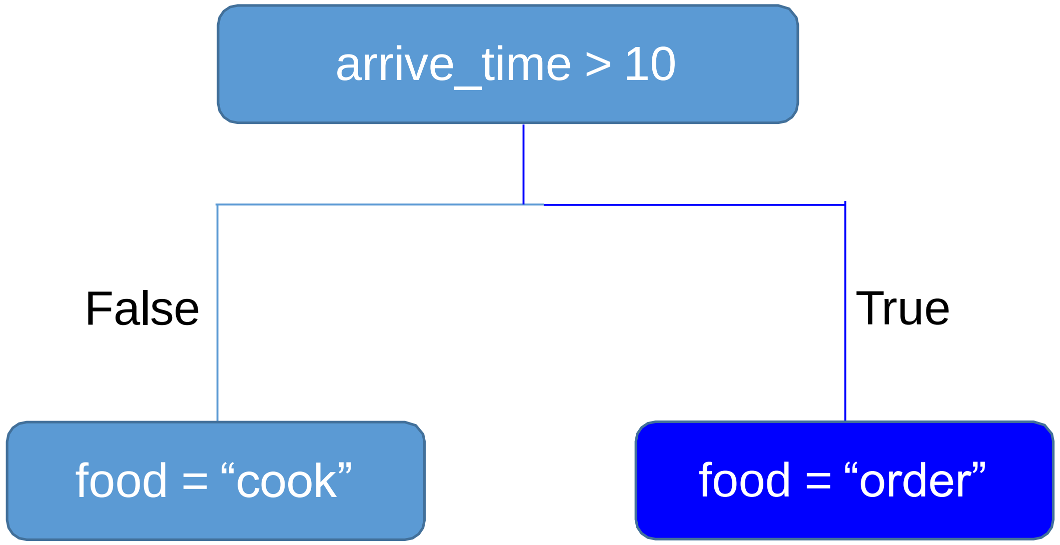


Conditional Statements

Pseudo-Code

```
Check if arrive_time > 10

then food = “order”
```



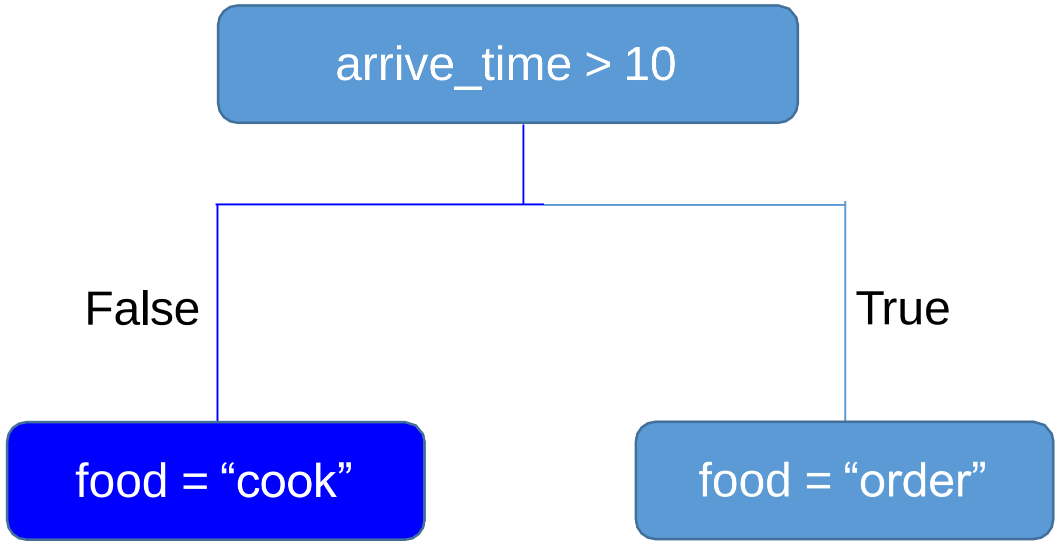
Conditional Statements

Pseudo-Code

```
Check if arrive_time > 10

then food = “order”

else food = “cook”
```



Conditional Statements

Pseudo-Code

```
Check if arrive_time > 10

then food = “order”

else food = “cook”
```

Code

Conditional Statements

Pseudo-Code

```
Check if arrive_time > 10

then food = “order”

else food = “cook”
```

Code

```
if arrive_time >10:
    food = “order”
else:
    food = “cook”
```

Conditional Statements

Pseudo-Code

```
Check if arrive_time > 10

then food = "order"

else food = "cook"
```

Code

```
if arrive_time >10:
    food = "order"
else:
    food = "cook"
```

Conditional Statements

Pseudo-Code

```
Check if arrive_time > 10

then food = “order”

else food = “cook”
```

Code

```
if arrive_time >10:
    food = “order”
else:
    food = “cook”
```

4 space “indentation”

Conditional Statements

Pseudo-Code

```
Check if arrive_time > 10

then food = “order”

else food = “cook”
```

Code

```
if arrive_time >10:
    food = “order”
else:
    food = “cook”
```

Conditional Statements

Pseudo-Code

```
Check if arrive_time > 10

then food = “order”

else food = “cook”
```

Code

```
if arrive_time >10:
   food = “order”
else:
   food = “cook”
```

Conditional Statements

Pseudo-Code

```
Check if arrive_time > 10

then food = “order”

else food = “cook”
```

Code

```
if arrive_time >10:
    food = “order”
else:
    food = “cook”
```

Conditional Statements: The if statement

- If – else statements : Single Condition

Pseudo-Code

```
Check if arrive_time > 10

then food = “order”

else food = “cook”
```

Syntax

```
if condition:
   statement 1
else:
   statement 2
```

Conditional Statements: Multiple conditions

- If – elif – else statements : Multiple Conditions

Conditional Statements: Multiple conditions

- If – elif – else statements : Multiple Conditions

  Example:

  Assume a variable x, print “positive” if x is greater than 0, “zero” if x is equal to 0 or “negative” if x is less than 0.

Conditional Statements: Multiple conditions

- If – elif – else statements : Multiple Conditions

Example:

Assume a variable x, print “positive” if x is greater than 0, “zero” if x is equal to 0 or “negative” if x is less than 0.

Pseudo-Code

```
Check if x>0
if yes then print(“positive”)

Otherwise check if x==0
if yes then print(“zero”)

For every other situation just print(“negative”)
```

Conditional Statements: Multiple conditions

- If – elif – else statements : Multiple Conditions

Example:

Assume a variable x, print “positive” if x is greater than 0, “zero” if x is equal to 0 or “negative” if x is less than 0.

Pseudo-Code

```
Check if x>0
if yes then print(“positive”)

Otherwise check if x==0
if yes then print(“zero”)

For every other situation just print(“negative”)
```

Code

```
if x>0:
    print(“positive”)
elif x==0:
    print(“zero”)
else:
    print(“negative”)
```

Conditional Statements: Multiple conditions

- If – elif – else statements : Multiple Conditions

Example:

Assume a variable x, print “positive” if x is greater than 0, “zero” if x is equal to 0 or “negative” if x is less than 0.

Pseudo-Code

```
Check if x>0
if yes then print(“positive”)

Otherwise check if x==0
if yes then print(“zero”)

For every other situation just print(“negative”)
```

Code

```
if x>0:
    print(“positive”)
elif x==0:
    print(“zero”)
else:
    print(“negative”)
```

Conditional Statements: The if-elif-else

- If – elif – else statements : Multiple Conditions

Pseudo-Code

```
Check if x>0
if yes then print(“positive”)

Otherwise check if x==0
if yes then print(“zero”)

For every other situation just print(“negative”)
```

Syntax

```
if condition1:
   statement 1
elif
   condition2:
   statement 2
else:
   statement 3
```

Conditional Statements: Multiple elifs

- If – elif – else statements : Multiple Conditions

Syntax

```
if condition1:
   statement 1
elif condition2:
   statement 2
           .
           .
           .
elif condition99:
   statement 99
else:
     statement 100
```

Looping Statements

```
for i in range(10):
    print(i)
```

```
print("hello")
print("world")
```

```
print("hello",end=" ")
print("world")
```

```
print("hello",end="*")
print("world",end="*")
print("to all...")
```

```
help(range)
```

```
range(10)
```

```
for i in range(10,100):
    print(i, end=" ")
```

```
a = 10
```

```
for i in range(100,10,-5):
    print(i, end=" ")
```

```
 100 95 90 85 80 75 70 65 60 55 50 45 40 35 30 25 20 15
```

```
mark_list = [95, 90, 85, 80, 75]

for mark in mark_list:
    if mark > 80:
        print("pass")
    else:
        print("fail")
```

```
 pass
 pass
 pass
 fail
 fail
```

```
subject = ['maths', 'science', 'social', 'english', 'tamil']
mark_list = [95, 90, 85, 80, 75]

for i in range(5):
    if mark_list[i] > 80:
        print(subject[i], mark_list[i]," - pass")
    else:
        print(subject[i], mark_list[i]," - fail")
```

```
 maths 95  - pass
 science 90  - pass
 social 85  - pass
 english 80  - fail
 tamil 75  - fail
```

```
for i in range(2,5):
    for j in range(1,5):
        print(i,j)
```

Show hidden output

```
number = int(input("enter a number"))
string = "Nielit Chennai"
for i in range(len(string)):
    print(i)
```

Show hidden output

```
for i in string:
    print(i)
```

Show hidden output

```
i = 0

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

Show hidden output

```
break, continue
```

```
i = 0

while i<10:
    if i==4:
        break
    else:
        print(i)
    i += 1
```

```
0
1
2
3
```

```
i = 0

while i<10:
    if i%2==0:
        i += 1
        continue
    else:
        print(i)
    i += 1
```

```
1
3
5
7
9
```

Conditional Statement

```
a = 1
b = 2
c = b-a
```

```
a is c
```

```
True
```

```
a == c
```

```
True
```

```
id(a)
```

```
140031922116912
```

```
id(c)
```

```
140031922116912
```

```
list1 = [1,2,3]
list2 = [1,2,3]
list3 = list1
```

```
list1 is list2
```

```
False
```

```
list1 is list3
```

```
True
```

```
list1 == list2
```

```
True
```

```
1 in list1
```

```
True
```

```
4 in list1
```

```
False
```

```
# syntax
# if condition:
    # statement
```

```
arrive_time = float(input("enter the arriving time : "))

if arrive_time > 10:
    food = "order"

else:
    food = "cook"

print(f"{food} your food")
```

```
enter the arriving time : 10
cook your food
```

if elif else statement

```
x = 2.5
```

```
if x>0:
    print(f"{x} is a positive number")
```

```
elif x==0:
    print(f"{x} is equal to zero")
else:
    print(f"{x} is a negative number")
```

```
 2.5 is a positive number
```

nested if statement

```
x = "-25"

if type(x)==int or type(x)==float:
    if x>0:
        print(f"{x} is a positive number")
    elif x==0:
        print(f"{x} is equal to zero")
    else:
        print(f"{x} is a negative number")
else:
    print("only integer and float numbers are accepted as input")
```

```
 only integer and float numbers are accepted as input
```

```
x = "hello"
type(x)
```

```
 str
```

```
type(25)==int
```

```
 True
```

For More Details

https://data-flair.training/blogs/python-operator/

Thank You