NUS AI SUMMER EXPERIENCE 2019

CONTROL STATEMENTS AND LOOPS

PAN BINBIN

NATIONAL UNIVERSITY OF SINGAPORE
DEPARTMENT OF ISEM

OUTLINE

01 CONTROL STATEMENTS

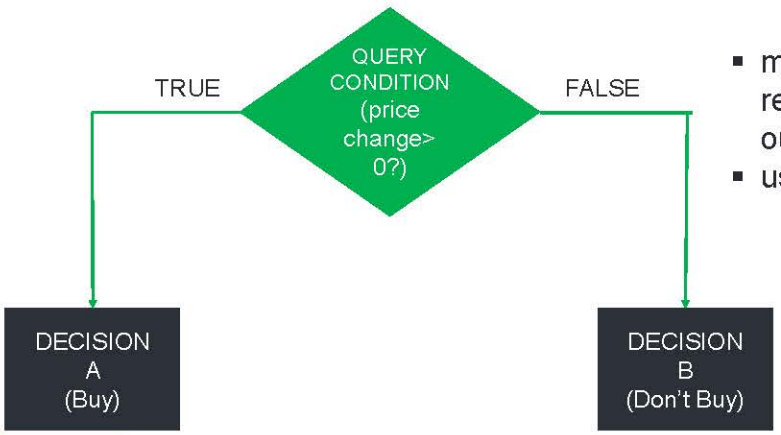
02 WHILE LOOPS

03 FOR LOOPS

NATIONAL UNIVERSITY OF SINGAPORE
DEPARTMENT OF ISEM

01 CONTROL STATEMENTS

COMPARISON AND DECISION



- most programming tasks requires actions based on outcome of comparison
- use of IF statements

- explicitly define input is a floating number

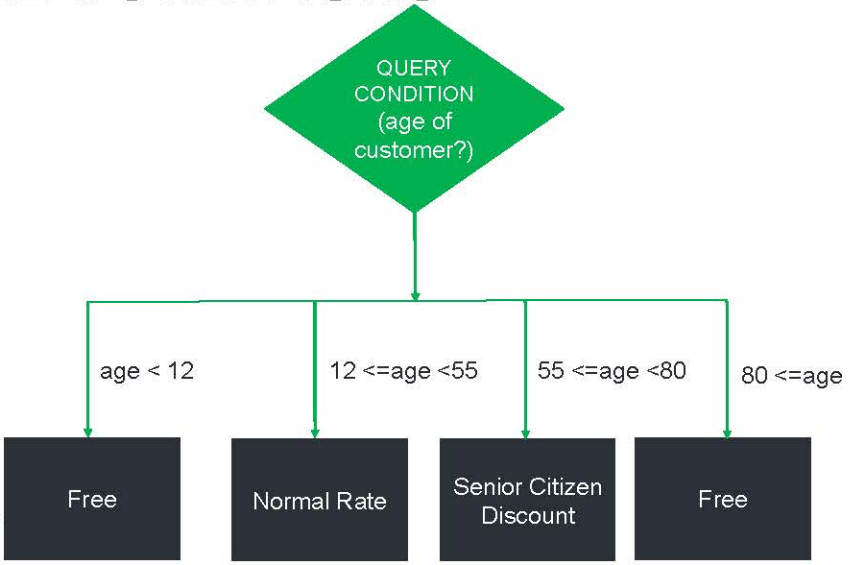
```
priceChange = float(input('What was the change in price: '))
if priceChange > 0:
    print('Buy')
else:
    print("Don't Buy")
```

```
What was the change in price: 10
Buy
```

- if statement used to compare
- if outcome is true, execute statement within the if –block
- Note the syntax
  - the if statement conclude with colon (:)
  - the statements within the if-block is indented with a space

- The statement under the else block is executed if the 'if' condition is false
- Note the quotes used in the statement in this else-block

MULTIPLE CONDITIONS



NATIONAL UNIVERSITY OF SINGAPORE
DEPARTMENT OF ISEM

```
age = int(input('Enter age of customer: '))
if age < 12 :
    print('Free')
elif age < 55 :
    print ('Normal Rate')
elif age < 80 :
    print ('Senior Citizen Discount')
else:
    print ('free')
```

```
Enter age of customer: 90
 free
```

- Note that the first elif-block should have two conditions (age > =12 and age < 55) but only test for one condition (age <55)
- Would those under 12 be charged normal rate (since they fulfil the condition age < 55
- No. Those under 12 will have fulfilled the if condition in the if-block above. The if-block will be executed and the program will never reach this elif block.
- Hence, there is no need to explicitly test for the age >=12 condition in this elif block. age must be >= 12 for program to reach this block
- Same argument holds for the other elif block

- There are three if, elif, blocks in addition to the else blocks
- The code check each block in the order they are written
  - the if-block first
  - the elif age <55 second
  - the elif age <80 third
- The first block with the condition being tested true is executed. Program is then exited.
- If none of the if, or elif block's condition is true, the else block is executed

NESTED IF

- Several if statements nested together

```
if condition I :
        if condition II :
                statements    #executed if condition I true AND condition II true
        else :
                statements    #executed if condition I true AND condition II false
else :
        statements            #executed if condition I false
```

Outer if-else block

Test if age >=12

If false, 'Free'

If true, go into the other if block

```
age = int(input('Enter age of customer: '))
if age >=12 :
    if age >= 80 :
        print('free')
    else:
        if age >= 55:
            print ('Senior Citizen Discount')
        else :
            print ('Normal Rate')
else:
    print('Free')
```

```
Enter age of customer: 47
Normal Rate
```

come to this innermost if-else block given the outcome of following conditions

age >=12 is true

age >= 80 is false

COMPARISON AND LOGICAL OPERATORS

- Operators for condition testing includes the following
  - comparison between variables
    - <, <=, ==, !=, >, >=
  - Membership checking
    - in, not in
  - Simple comparison combined by logical operators
    - and, or, not

EXAMPLE OF USE OF LOGICAL OPERATORS

Logical operator 'or' used

syntax is: age < 12 or age >= 80

error occur if you write it as age < 12 or >=80 (without the second 'age')

```
age = int(input('Enter age of customer: '))
if age < 12 or age >= 80 :
    print('Free')
elif age >=55 :
    print ('Senior Citizen Discount')
else :
    print ('Normal Rate')
```

```
Enter age of customer: 20
Normal Rate
```

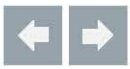

EXAMPLE OF MEMBERSHIP CHECK

Whether the word 'John' appears in the sentence

```
sentence = input()
print (sentence)
if 'john' in sentence:
    print('send to john')
else:
    print('return to sender')
```

```
john is a good boy
john is a good boy
send to john
```

02 WHILE LOOPS



WHAT ARE LOOPS

- Statements that get executed over and over again as long as some conditions a
- Inputs to the statements can differ for each cycle that statements are run
- Control point:
  - while condition (true of false)
  - for condition (as long as there is still something in sequence)
  - deliberate stop
    - break
    - continue

NATIONAL UNIVERSITY OF SINGAPORE
DEPARTMENT OF ISEM

WHILE LOOP

- General Format

  while condition :

  statements

- Syntax

  colon (:) at end of while expression

  indent for statements after while expression but part of the while block

- Condition is a Boolean expression. Evaluated as True or False
- If condition is true, all statements are executed once.
- Condition is evaluated again. If True, repeat execution of statement
- If false, program exits the while loop

EXAMPLE OF WHILE LOOP

Let's say you want to add a number from 1 to n.
Ask for input to n

- initialize variables
- i - the number to be added
- total – store the sum

- the while expression
- n is the last number to add
- as long as (i <=n) is True, the statements in the while loop will be executed

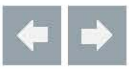
```
n = int(input('Enter the integer that you want to sum to: '))
i = 0
total = 0
while i <=n:
    total += i
    i += 1
print (total)
```

- current value of i is added to total

- current value of i is increased by 1
- last statement in the while block
- after this, it goes back to the while expression to test whether (i <= n) is true.
- Since i is increased at each iteration, there will come a point when (i <= n) is false and the statements in the while block is not executed anymore

- this print statement is outside the while block
- absence of indent
- it is executed after the program exits the while block

03 FOR LOOPS



FOR LOOP

- General Format

  for item in sequence :

  statements #usually related to item

- Syntax

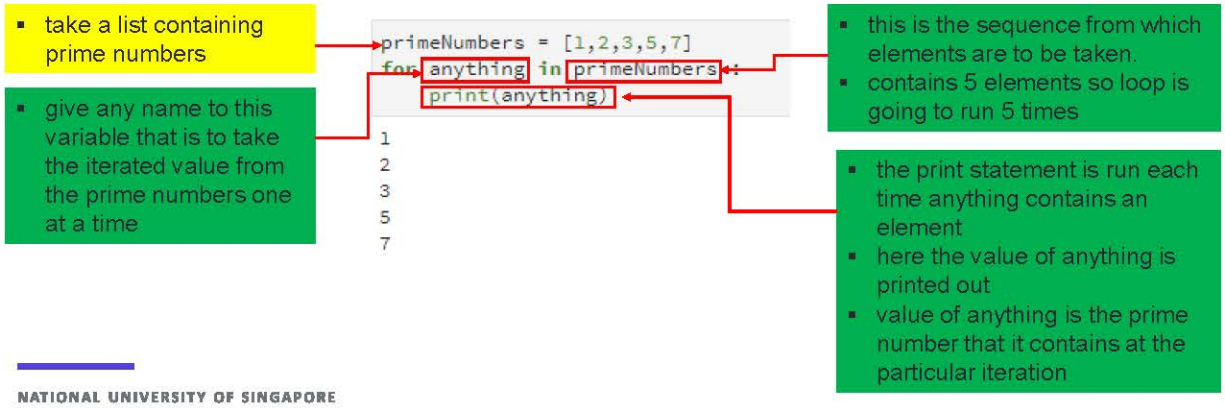
  colon (:) at end of for expression

  indent for statements after while expression but part of the while block

- Sequence can be list, string, dictionary
- item is just a variable name and not a special keyword in Python
- item takes each element from sequence in the order that they appear in sequence.
- once item takes an element, the statements in the for loop is executed
- execution of statements in the for loop stops once the sequence in empty.

NATIONAL UNIVERSITY OF SINGAPORE
DEPARTMENT OF ISEM

EXAMPLE OF FOR LOOP

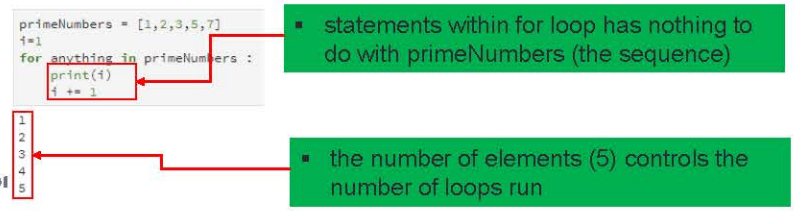


EXAMPLE OF FOR LOOP STATEMENTS

The statements inside the for-loop does not need to have anything to do with the item that carries the value from the sequence:

for item in sequence

The example below illustrates this
However, it usually does because the purpose of the for loop is to operate on the sequence



BREAK COMMAND

- Terminates the execution of a while-loop or a for-loop once it is encountered
- Efficiency of program
- Syntax

```
for item in sequence :
        statements
        if condition :
                statements    #statements to be executed if condition true
                break         # exit the for-loop
else                          #else here is part of for –loop, not part of if statement. Optional.
                              #used with break command.
                              #executed only when break is not executed
statements
```


Example of break COMMAND

- Let's say you want to check whether a given number is a prime number.
- An easy test is to determine whether the given number if divisible by another number
- Just try dividing the given number by numbers smaller than itself/2. Start with 2 and increase the divisor by 1 each time
- Once a divisor is found, we know that given number is not a prime number
- Can stop the test. Use the break command

EXAMPLE: Prime Number

- test all numbers smaller than n as divisors
- range (2,n) will generate a list of integers from 2 to n-1
- assigns generated integer to i, one per loop

- testing whether current i is a divisor
- % is a modulo operator

- if n%i == 0 is true
- print out information
- break out of the for loop
- if n%i == 0 false, program returns to for expression and gets next i

- if n%i == 0 is never true for all i in range (2,n)
- no divisor found
- break is not executed
- else will explain why break is not executed
- else will not be executed if break is executed

```
n= int(input('Enter the integer you wish to test whether is a prime number: '))
for i in range(2,n):
    if n%i == 0:
        print(n, 'is divisible by',i)
        print(n, 'is not a prime number')
        break
else:
    print (str(n) + ' is a prime number')
```

```
125 is divisible by 5
125 is not a prime number
```


CONTINUE COMMAND

- Ends current iteration of a while-loop or a for-loop once it is encountered
- Go directly to next iteration on the loop
- Efficiency of program
- Syntax

```
for item in sequence :
    statements
    if condition :
        statements    #statements to be executed if condition true
        continue          # exit the current iteration
                          #ignore statements below the continue command
    statements
```

EXAMPLE

- Let's say you want to print out all integers from 1 to n that satisfies the two conditions together.
  - condition 1: has remainder of 1 when divided by 2; and
  - condition 2: has remainder of 2 when divided by 3
- Carry out the test for the conditions one at a time
- If an integer does not satisfy condition 1, there is no need to test for condition 2.
- Use continue statement and move on to the next iteration. Test the next number

Example of continue COMMAND

- range (1,n+1) will generate a list of integers from 1 to n

- i%2 != 1 will be True if the remainder is not 1
- condition 1 (remainder of number divided by 2 is 1) is not true
- continue stops iteration and goes back to the top of the for-loop
- all statements below for loop not executed in this iteration

- the print statement is executed only if none of the two continue statements above are not executed
- both conditions satisfied

```
n= int(input('Enter the integer you wish to test up to: '))
for i in range(1,n+1):
    if i%2 !=1:
        continue
    if i%3 != 2:
        continue
    print(i, end = ' ')
```

```
Enter the integer you wish to test up to: 30
 5 11 17 23 29
```

PAN BINBIN

BINBIN.PAN@NUS.EDU.SG

NUS National University of Singapore

NUS AI SUMMER EXPERIENCE 2019

NATIONAL UNIVERSITY OF SINGAPORE
DEPARTMENT OF ISEM

CONTROL STATEMENTS AND LOOPS