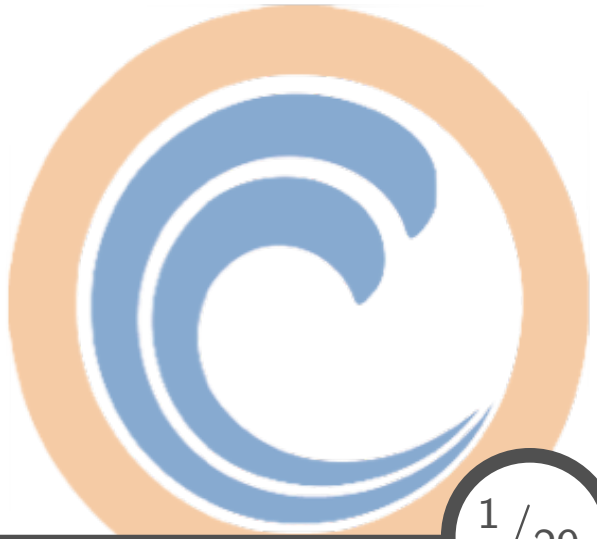
CS A131: Lecture 5

Nadia Ahmed

Orange Coast College, Computer Science

CS1A



Overview

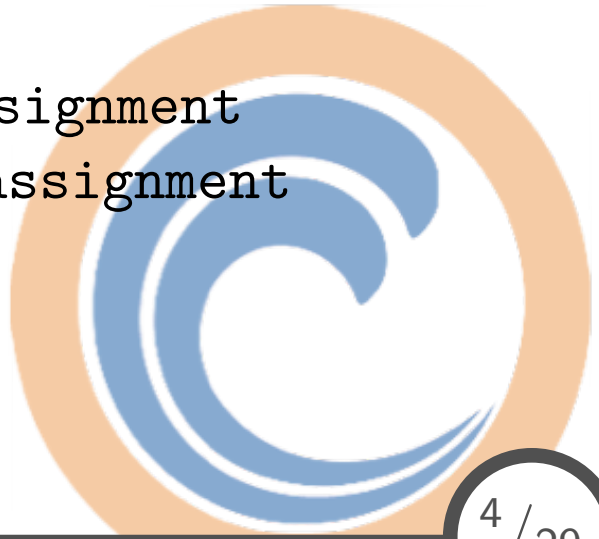
- Counters
  - Augmented Assignment Operators
  - Increment and Decrement Operators
- Repetition Statements
  - `while` loop
  - `for` loop
- Counter-controlled repetition
  - Example `average.py`
- Sentinel-controlled repetition
  - Example `average2.py`

Overview

- Think!
- Structured Programming
  - Control flow charts
  - Sequential statements
  - Conditional statements
    - `if` statement
    - `if-else` statement
    - `if-elif-else` statement
  - Repetition statements
    - `while` loop
    - `for` loop
  - Example `interest.py`

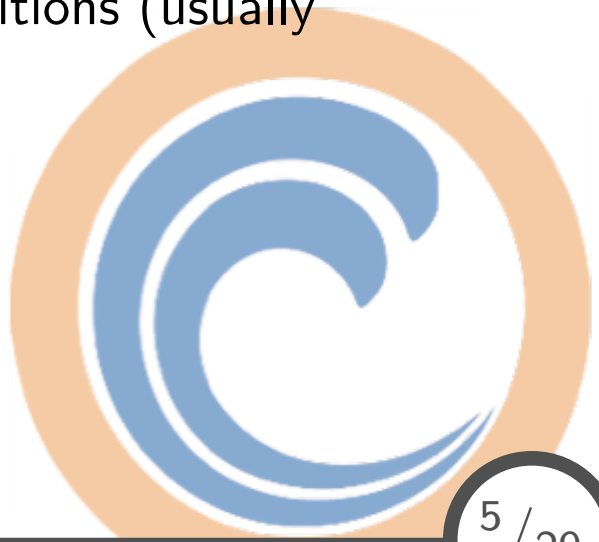
Augmented Assignment Operators

- Assignment operator: =
  - evaluates right-hand side
  - assigns result to left-hand side
- Augmented assignment operators: +=, *=, ...
  - evaluates right-hand side as temporary result
  - applies operation to left-hand side and temporary result
  - assigns result of operation to left-hand side
- Example: Counter
  - `c = 0  # counter starting from 0`
  - `c = c + 1  # counting by regular assignment`
  - `c += 1  # counting by augmented assignment`
- Augmented assignment operators:
  - +=, -=, *=, /=, %=, <<=, >>=

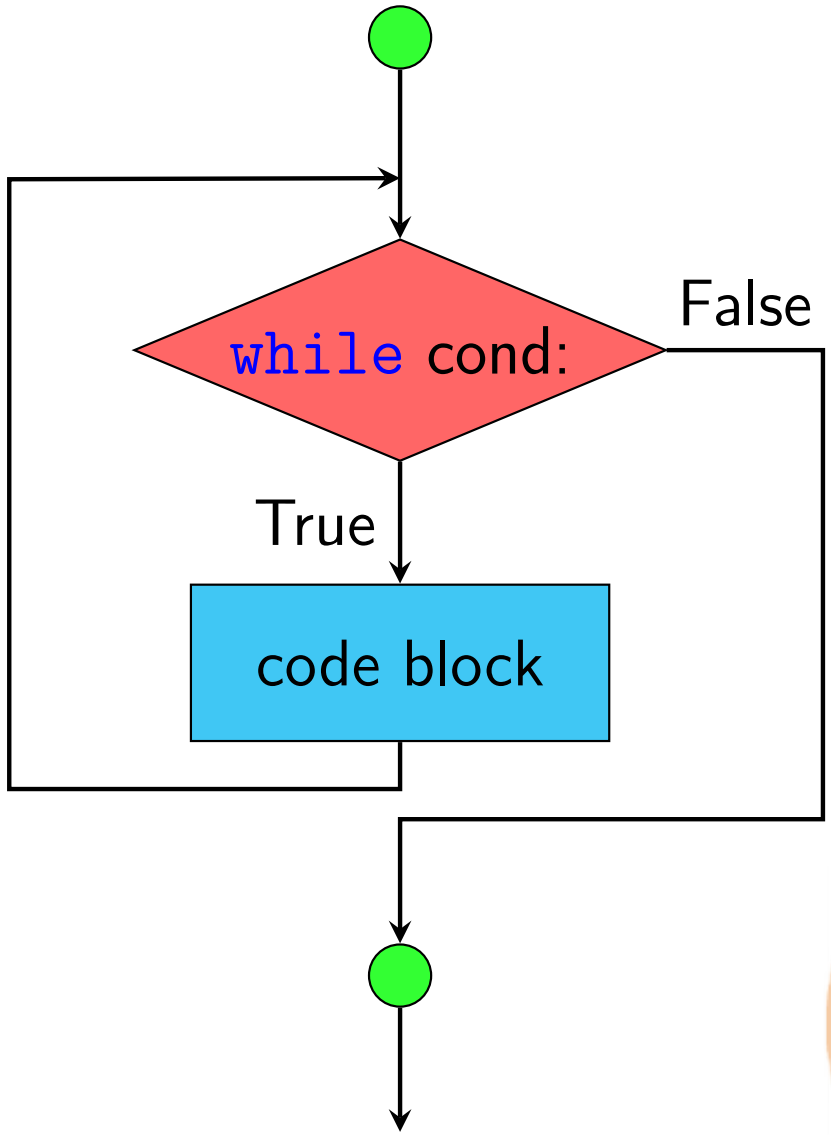


Repetition Statement

- Repetition
  - repeated execution of a block of statements
  - counter-controlled
    - counter determines number of repetitions (often predefined at compile time)
  - sentinel-controlled
    - sentinel condition determines number of repetitions (usually determined at run-time)



Control flow chart



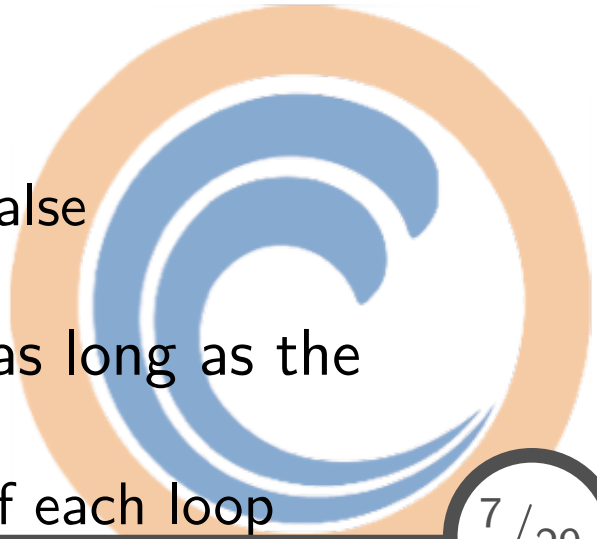
Repetition Statement

`while` loop

- Control flow statement for repetition (iteration)
  - Repeats execution depending on specified condition
  - All statements in the `while` block must be consistently indented for the interpreter to tell where the block begins and ends.
- Example

```
product = 2
while product < 1000:
  product *=2
#elihw
print( "Product is %d" % product)
```

- Syntax `while` construct consists of
  - keyword `while`
  - condition: expression evaluated to True or False
  - body statement block
- Semantics: the body is repeatedly executed as long as the condition evaluates to true
  - the condition is evaluated at the *beginning* of each loop



Counter Controlled Example: average.py

```
##############################################################
# average.py: compute the average of a set of numbers
# author: Nadia Ahmed
# modifications:
# 07/12/2016  NA initial version
##############################################################
def main():
  # input and computation section
  counter = 1;
  total = 0.0;
  # while loop
  while counter <= 10:
    value = int(input("Please␣enter␣a␣value:␣"))
    total += value
    counter += 1
  #elihw
  # computation section
  average = total/ 10.0
  # output section
  print("The␣average␣is:␣%f" % average)
# EOF
# function call
main()
```

Sentinel Controlled Example: average2.py

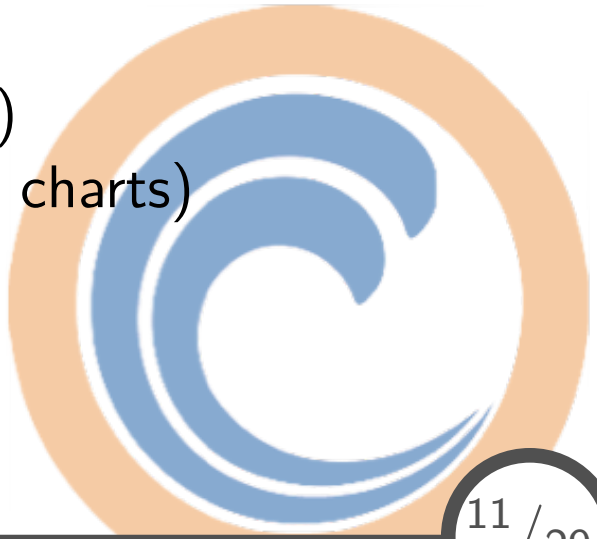
```
/#########################################################
# average2.py: compute the average of a set of numbers
# author: Nadia Ahmed
# modifications:
# 07/12/2016  NA sentinel controlled loop
# 07/12/2016  NA initial version
#########################################################/
def main():
   # variable initialization
   counter = 0
   total = 0.0

   # initialize the loop control variable value
   value = float(input("Please␣enter␣a␣value␣(or␣-1␣to␣quit):␣"))
   ...
```

Sentinel Controlled Example: average2.py

```
    ...
    while value != -1.0:  # check the loop control variable value
        total += value
        counter += 1
        # update the loop control variable value
        value = float(input("Please␣enter␣a␣value␣(or␣-1␣to␣quit):␣"))
    # elihw
    print("%d␣value␣entered." % counter)
    if counter >= 1:
      average = total/counter
      print("The␣average␣is␣%f" % average)
    # fi
#EOF
# function call
main()
```

Programming == Thinking

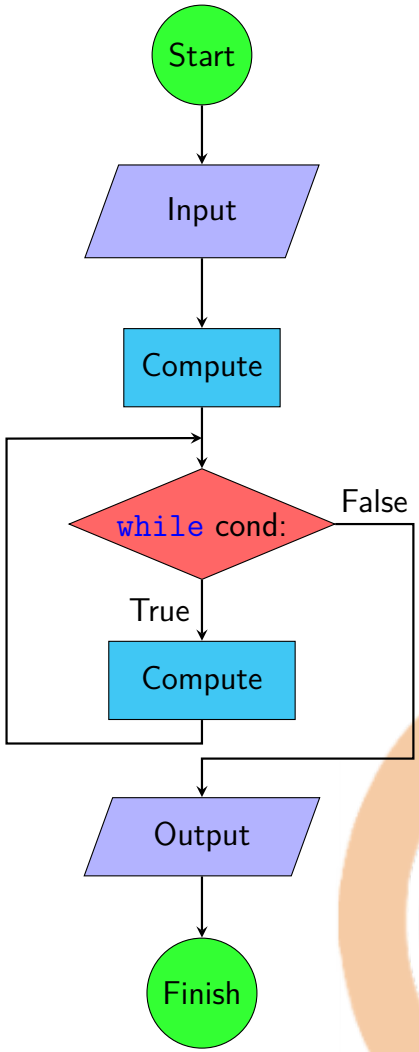
- Programming
  - *not* a mechanical procedure
  - requires *thinking*
- Program
  - *writing* requires human intelligence
  - *execution* can be done by a *dumb* machine.
- General programming steps
  1. Understand the problem
  2. Define the input and output data
  3. Develop the algorithm (e.g. use pseudo code)
  4. Define the control flow (e.g. use control flow charts)
  5. Write the program in programming language
  6. Test and debug the program



Control Flow Charts

Graphical representation of program control flow

- Sequential Execution
- Selection
- Termination

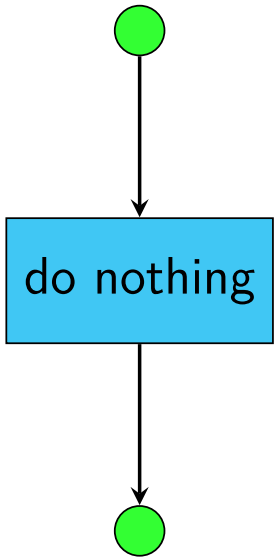


Structured Programming

Empty statement blocks

- empty compound statement
- does nothing (no operation, no-op)
- Example

```
# nothing
```

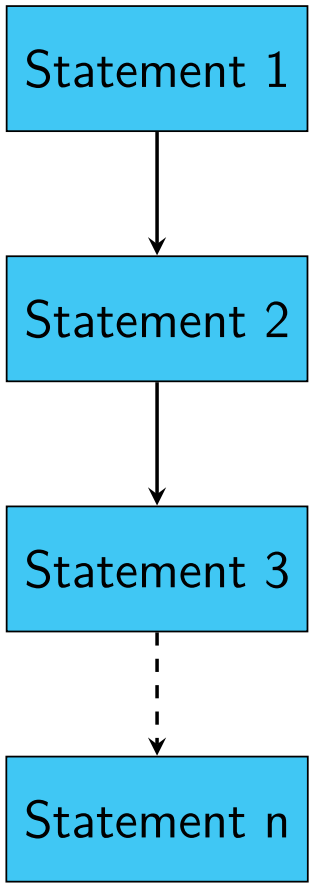


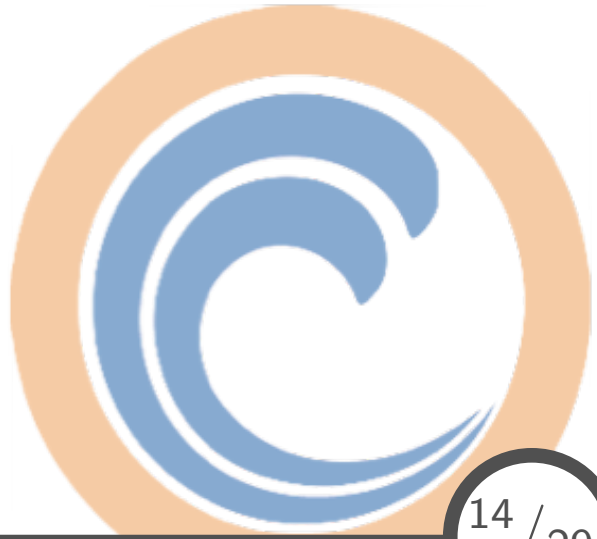
Structured Programming

Sequential Execution in Python

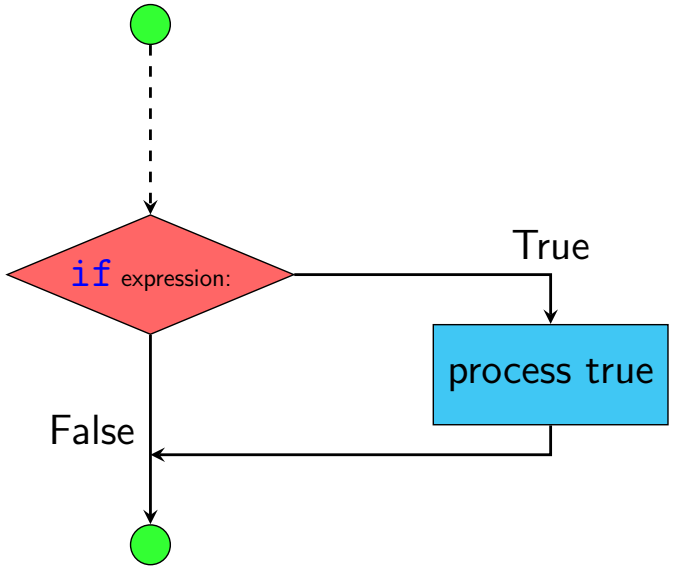
- Statement blocks: *Compound statements*

```
# statement 1


# statement 2


# statement 3

...

# statement n
```





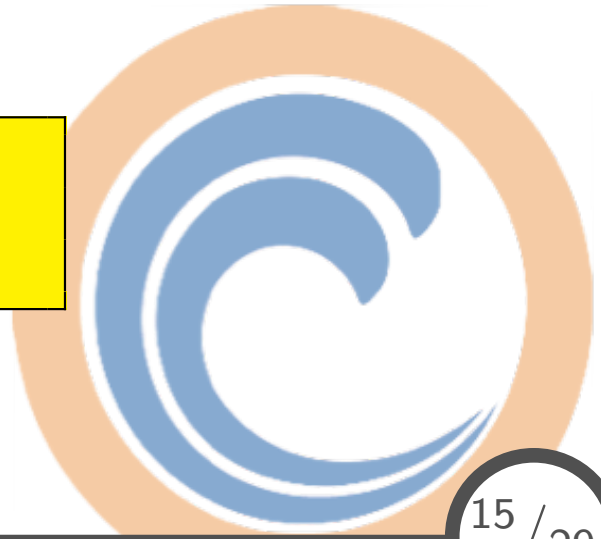
Selection: `if` statement



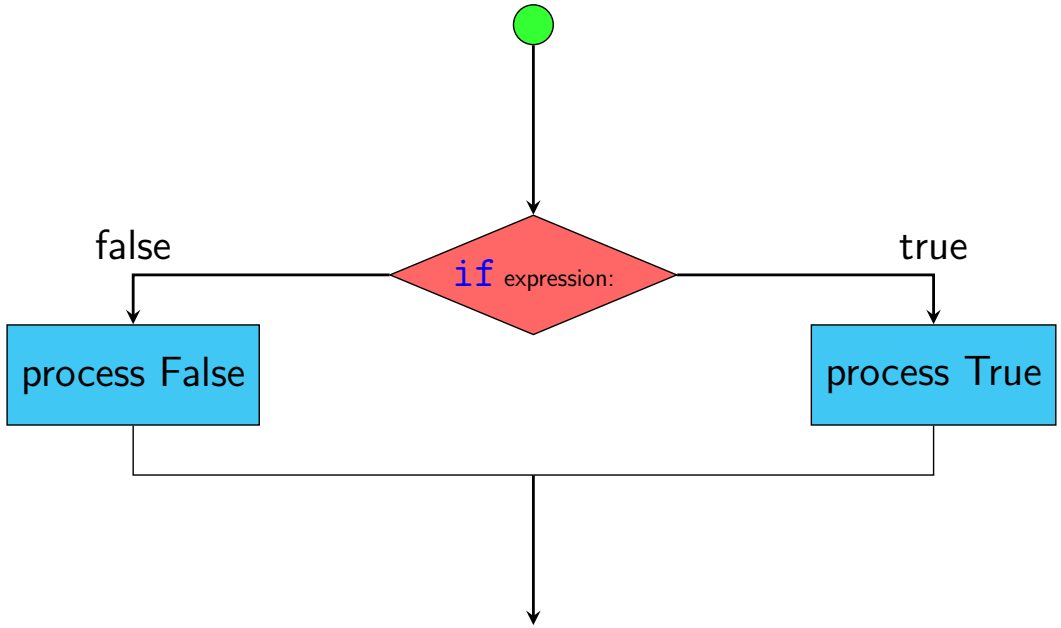
- Flow chart:

- Example:

```
if grade >= 60:
  print("You passed.")
# fi
```



Selection: `if-else` statement

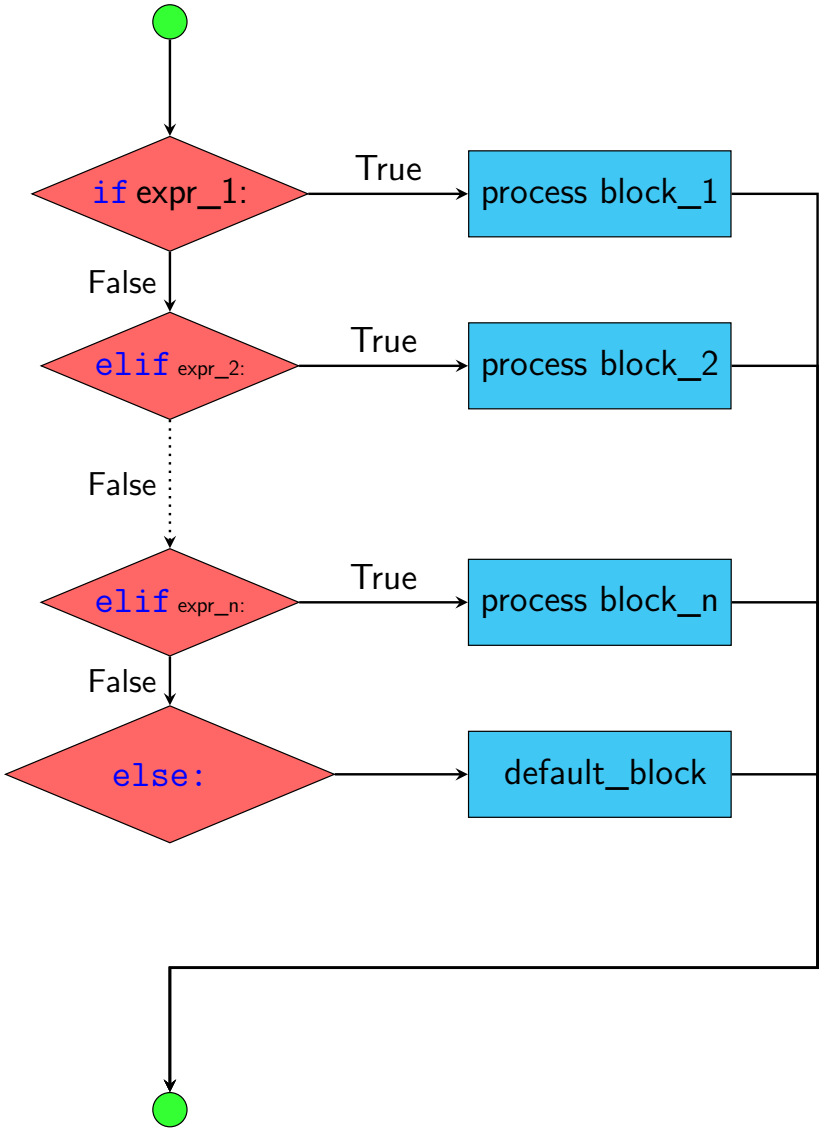


- Flow chart:
- Example:

```
if grade >= 60:
  print("You passed.")
# fi
else:
  print("You failed.")
# esle
```

Multiple Selections: `if else-if` statement

- Flow chart:

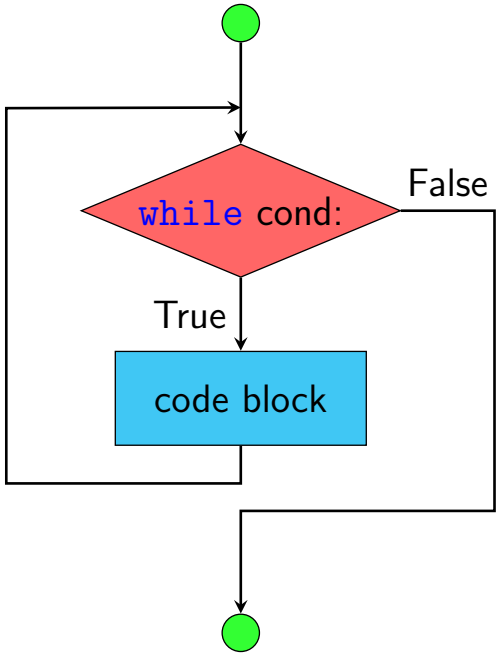


- Example:

```
if grade >= 90:
  print("Excellent!")
# fi
elif grade >= 80:
  print("Satisfactory.")
  .
  .
  .
# file
elif grade >=60:
  print("You␣passed.")
# file
else:
  print("Failed.")
# esle
```

Python Repetition: `while` loop

- Flow Chart:



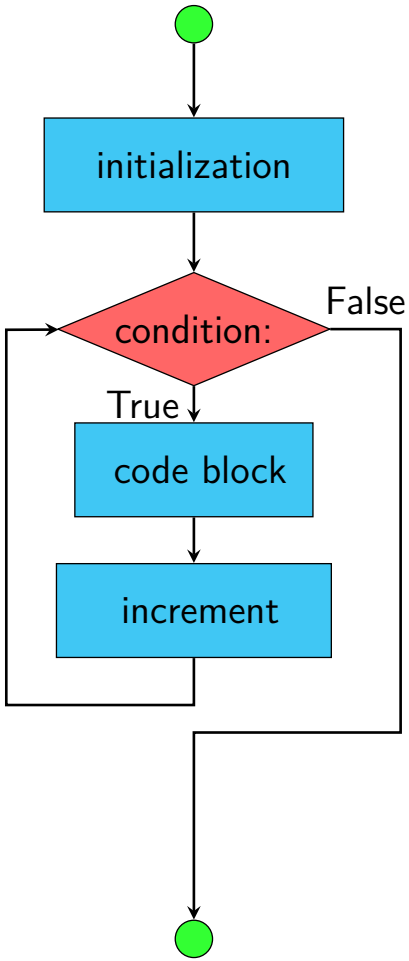
- Example:

```
product = 2;
while product < 1000:
  product *= 2;
# elihw
```

- Note: The condition is evaluated at the *beginning* of each loop! `while` loop is a *pretest* loop!

Python Repetition: `for` loop

- Flow Chart:



- Example:

```
for i in [0,1,2,3,4,5,6,7,8,9]:
  print("i = %d " % i)
# rof
```

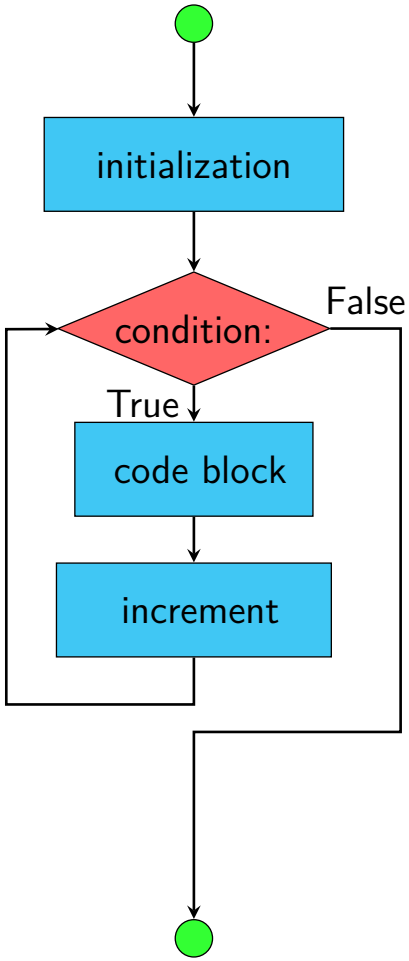
- Syntax:

```
for variable in [val1, val2,...]:
  # statement
  # statement
  # etc.
```

- Note: The condition is evaluated at the *beginning* of each loop! `for` is a *pretest* loop!

Python Repetition: `for` loop using `range()`

- Flow Chart:



- Example:

```
# range counts from 0 to one
# less than the input argument
for i in range(10):
  print("i = %d " % i)
# rof
```

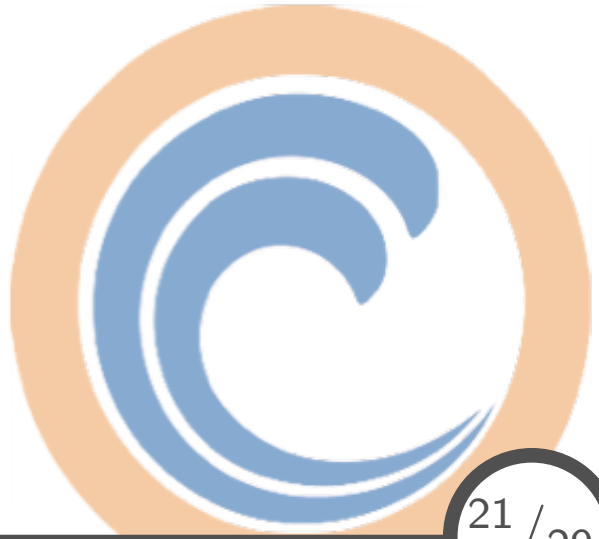
- Syntax:

```
for variable in range(bound):
  # statement
  # statement
  # etc.
```

- Note: The condition is evaluated at the *beginning* of each loop! `for` is a *pretest* loop!

Choosing the Best Loop based on Application

- If you can determine in advance the number of repetitions use the `for` loop or the counter controlled `while` loop
- If you do not know and cannot determine the number of repetitions needed, and it could be zero use a `while` loop

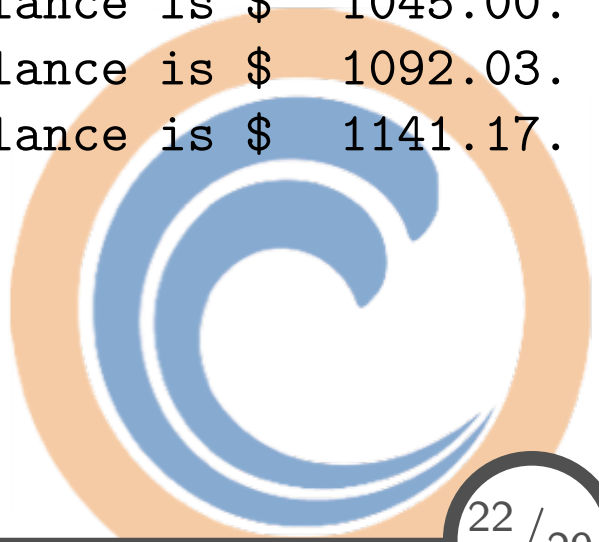


Example Program

Compound interest: `interest.py`

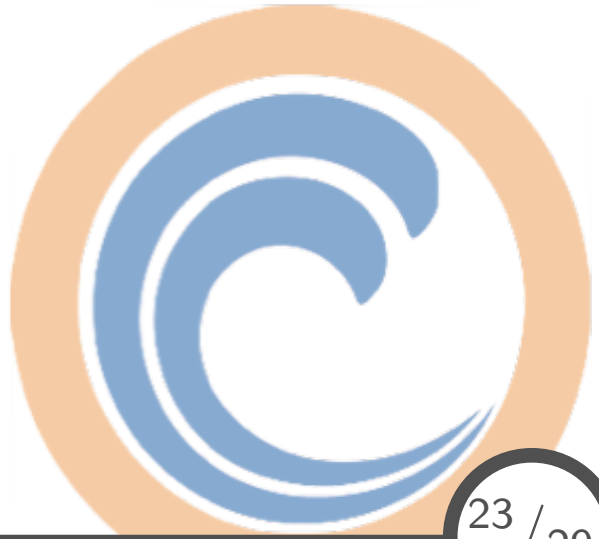
- Write a program that calculates the interest accumulated in a savings account. Given an initial deposit amount an an annual percentage rate (APR), compute the yearly interest earned and the resulting balance, for a period of ten years.
- the output should be listed in the table as follows:

```
Interest for year   1 is $   45.00, total balance is $   1045.00.
Interest for year   2 is $   47.02, total balance is $   1092.03.
Interest for year   3 is $   49.14, total balance is $   1141.17.
...
```



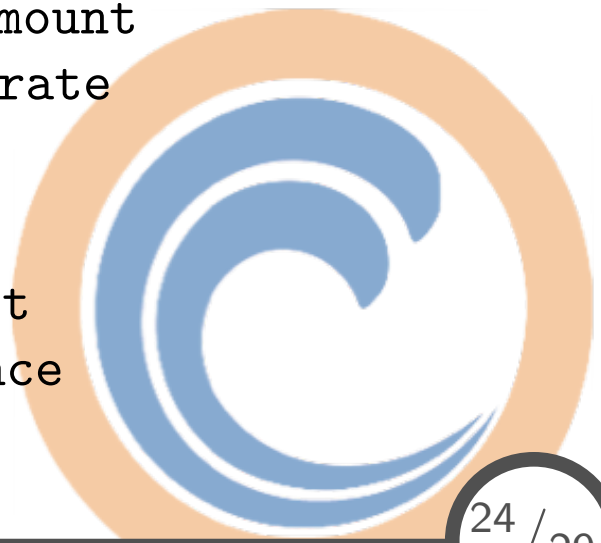
Example Program

- Compound interest: `interest.py`
- Assignment
  - Write a program that calculates the interest accumulated in a savings account. Given an initial deposit amount and an annual percentage rate (APR), compute the yearly interest earned and the resulting balance, for a period of ten years.
- Step 1: Understand the problem
  - What is given?
    - deposit amount
    - annual percentage rate
  - What is asked for?
    - yearly interest
    - resulting balance
  - How do we compute what is asked for?
    - interest = amount * APR/100
    - balance = amount + interest



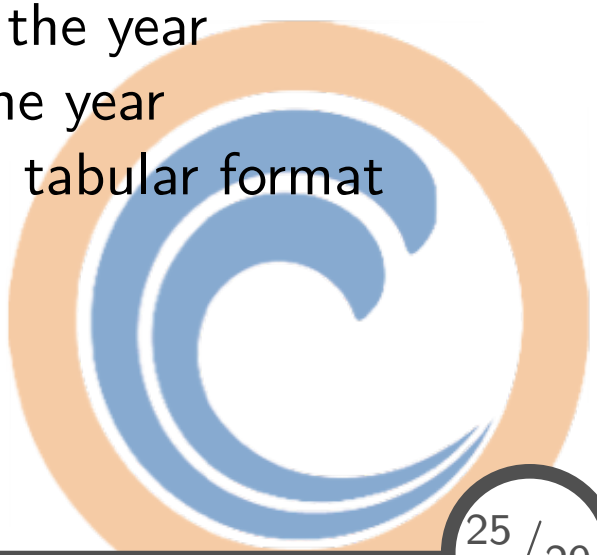
Example Program

- Compound interest: `interest.py`
- Assigment
  - Write a program that calculates the interest accumulated in a savings account. Given an initial deposit amount and an annual percentage rate (APR), compute the yearly interest earned and the resulting balance, for a period of ten years.
- Step 2: Define the input and output data
  - Input
    - Initial deposit amount: floating point value, `amount`
    - Annual percentage rate: floating point value, `rate`
  - Output
    - Current year: integral value, `year`
    - Interest earned: floating point value, `interest`
    - Resulting balance: floating point value, `balance`



Example Program

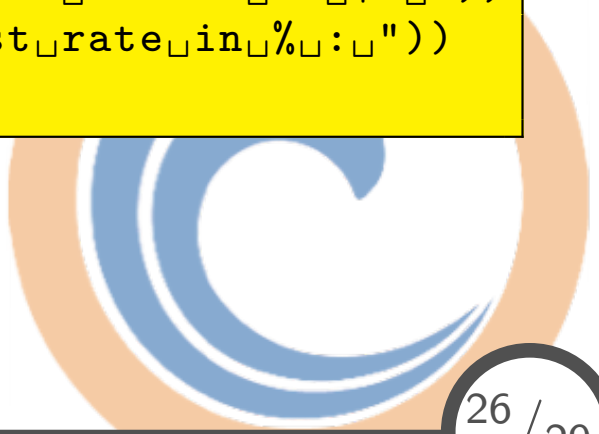
- Compound interest: `interest.py`
- Assignment
  - Write a program that calculates the interest accumulated in a savings account. Given an initial deposit amount and an annual percentage rate (APR), compute the yearly interest earned and the resulting balance, for a period of ten years.
- Step 3: Develop the algorithm
  - First, input `amount` and `rate`
  - Next, compute `interest` on the `amount` for the year
  - Next, compute new `balance` at the end of the year
  - Then, print `year`, `interest` and `balance` in tabular format
  - Finally, set the `amount` to the new `balance`
  - Repeat the previous 4 steps for 10 years
  - Done!



Example Program

- Compound interest: `interest.py`
- Assignment
  - Write a program that calculates the interest accumulated in a savings account. Given an initial deposit amount and an annual percentage rate (APR), compute the yearly interest earned and the resulting balance, for a period of ten years.
- Step 4: Write the program in programming language

```
...
amount = float(input("Please␣enter␣the␣initial␣amount␣in␣$:␣"))
rate = float(input("Please␣enter␣the␣interest␣rate␣in␣%␣:␣"))
...
```



Example Program: interest.py part (1/2)

```
/#########################################################
# interest.py: compound interest on savings account
# author: Nadia Ahmed
# modifications:
# 07/19/2016 NA distinguish amount and balance
# 07/20/2016 NA initial version
#########################################################/

def main():
  # input section
  amount = float(input("Please␣enter␣the␣initial␣amount␣in␣$:␣"))
  rate = float(input("Please␣enter␣the␣interest␣rate␣in␣%:␣"))
  ...
```

Example Program: interest.py part (2/2)

```
...
  # computation and output section
  for year in range(1,11):
    interest = amount * (rate/100.0);
    balance = amount + interest;
    print("Interest␣for␣year␣%d␣is␣$␣%.2f,␣total␣balance␣is␣$␣%.2f
        ." % (year, interest, balance))
    amount = balance;
  # rof
main()
```

Example Program

- Compound interest: `interest.py`
- Assignment
  - Write a program that calculates the interest accumulated in a savings account. Given an initial deposit amount and an annual percentage rate (APR), compute the yearly interest earned and the resulting balance, for a period of ten years.
- Step 6: Test (and debug) the program