Elements of programs

CHAPTER 2

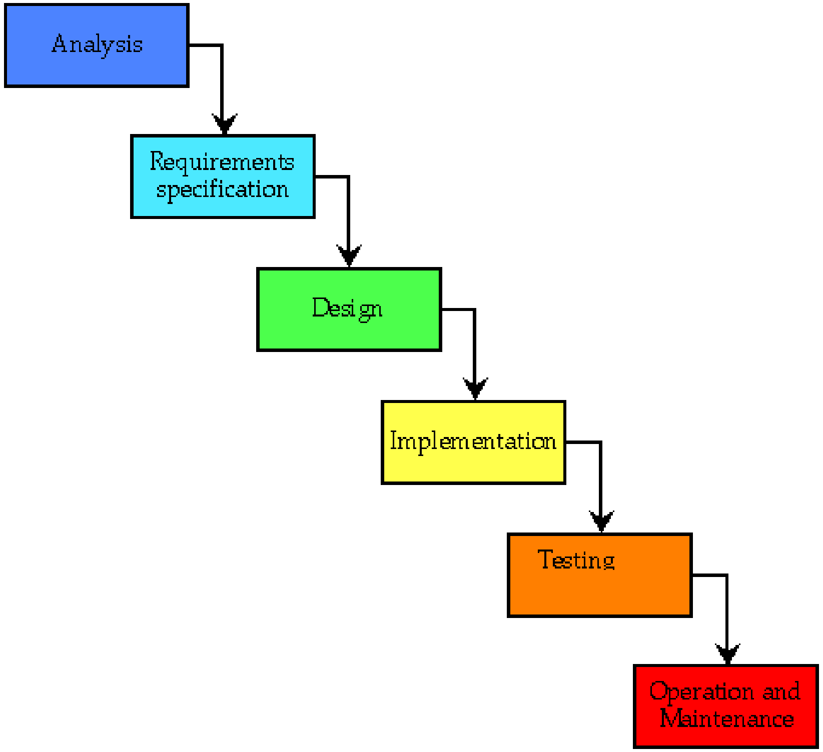
Objectives

To be able to:

- Understand and apply the software development process
- Produce pseudo code and flowchart
- Understand the element of program
- Get numeric information entered from the keyboard and output information

The Software Development Process

The process of creating a program is often broken down into stages according to the information that is produced in each phase.



The Software Development Process

Analyze the Problem

Figure out exactly the problem to be solved.

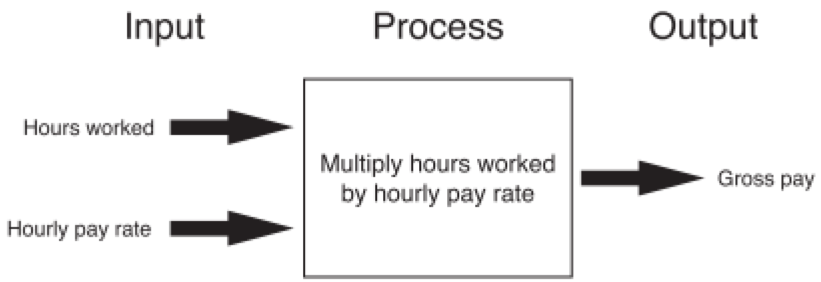
Try to understand it as much as possible.

The Software Development Process

Determine Specifications

Describe exactly what your program will do.

- Don’t worry about *how* the program will work, but *what* it will do.
- Includes describing the inputs, outputs, and how they relate to one another.



The Software Development Process

Create a Design

- Formulate the overall structure of the program.
- This is where the *how* of the program gets worked out.
- You choose or develop your own algorithm that meets the specifications.

The Software Development Process

Implement the Design

- Translate the design into a computer language.
- In this course we will use Python.

The Software Development Process

Test/Debug the Program

- Try out your program to see if it worked.
- If there are any errors (*bugs*), they need to be located and fixed. This process is called *debugging*.
- Your goal is to find errors, so try everything that might “break” your program!

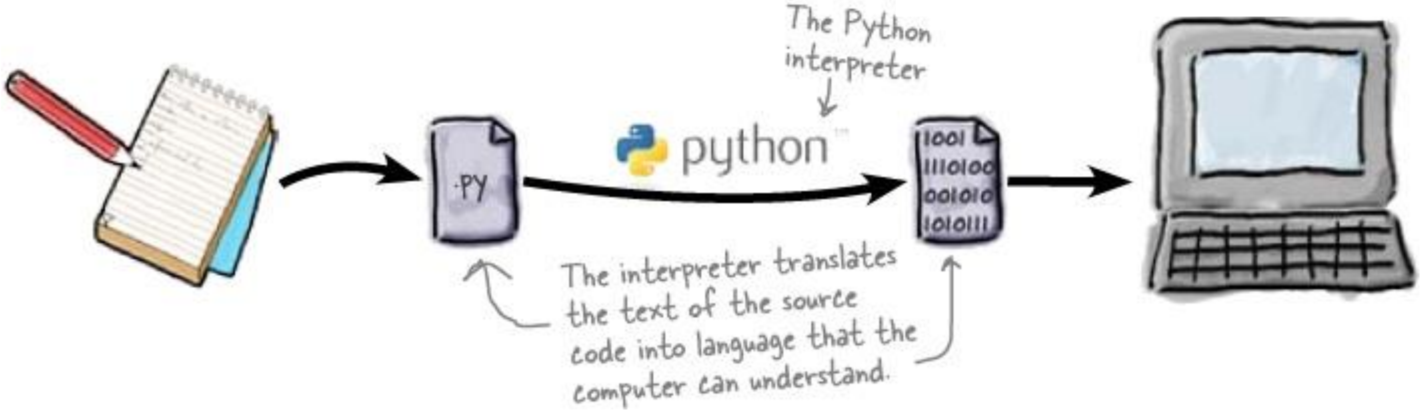
The Software Development Process

Maintain the Program

- Continue developing the program in response to the needs of your users.
- In the real world, most programs are never completely finished – they evolve over time.

phillipmartin.info

Example: Temperature Converter


The Python interpreter
.py
python
1001
1110100
001010
1010111
The interpreter translates the text of the source code into language that the computer can understand.

Example Program: Temperature Converter

Analysis – the temperature is given in Celsius, user wants it expressed in degrees Fahrenheit.

Example Program: Temperature Converter

Specification

- Input – temperature in Celsius
- Output – temperature in Fahrenheit
- **Process**
- Output = 9/5(input) + 32

Example Program: Temperature Converter

Design

- Input, Process, Output (IPO)
- Prompt the user for input (Celsius temperature)
- Process it to convert it to Fahrenheit using F = 9/5(C) + 32
- Output the result by displaying it on the screen

Pseudocode

- Natural language-like statements
- Statement describe action
- Focus on logic of program
- Steps are numbered
- Indentation used for dependent statement (i.e. selection and repetition)

Pseudocode Language Construct

Computation/Assignment

- Compute x as the multiplication of a and b
- Increment counter by 1
- Assign 0 to y

Input/Output

- Get a and b
- Display a and b

Pseudocode Language Construct

Selection: If … else

Single-selection

```
1. IF condition THEN
   1.1 statement 1
   1.2 statement 2
```

Double-selection

```
2. IF condition THEN
   2.1 statement 1
   2.2 etc
3. ELSE
   3.1 statement 1
```

Pseudocode Language Construct

Repetition: while

```
1. WHILE condition
   1.1 statement 1
   1.2 etc
```

Repetition: do…while

```
1. DO
   1.1 statement 1
   1.2 etc
2. WHILE condition
```

Repetition: for

```
1. FOR repetition criteria
   1.1 statement 1
   1.2 etc
```

Example Program: Temperature Converter

Before we start coding, let's write a rough draft of the program in *pseudocode*

Pseudocode is precise English that describes what a program does, step by step.

Using pseudocode, we can concentrate on the algorithm rather than the programming language.

Example Program: Temperature Converter

Pseudocode:

```
Input the temperature in degrees Celsius (call it celsius)
Calculate fahrenheit as (9/5)*celsius+32
Display fahrenheit
```

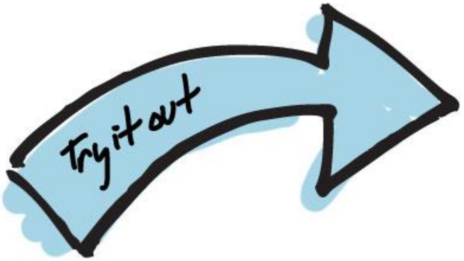
Each statement represents an operation

Now we need to convert this to Python!

Pseudocode Example

Programming requirements:

Express an algorithm to get two numbers from the user(dividend and divisor), testing to make sure that the divisor number is not zero, if it is, ask user to input again otherwise display their quotient

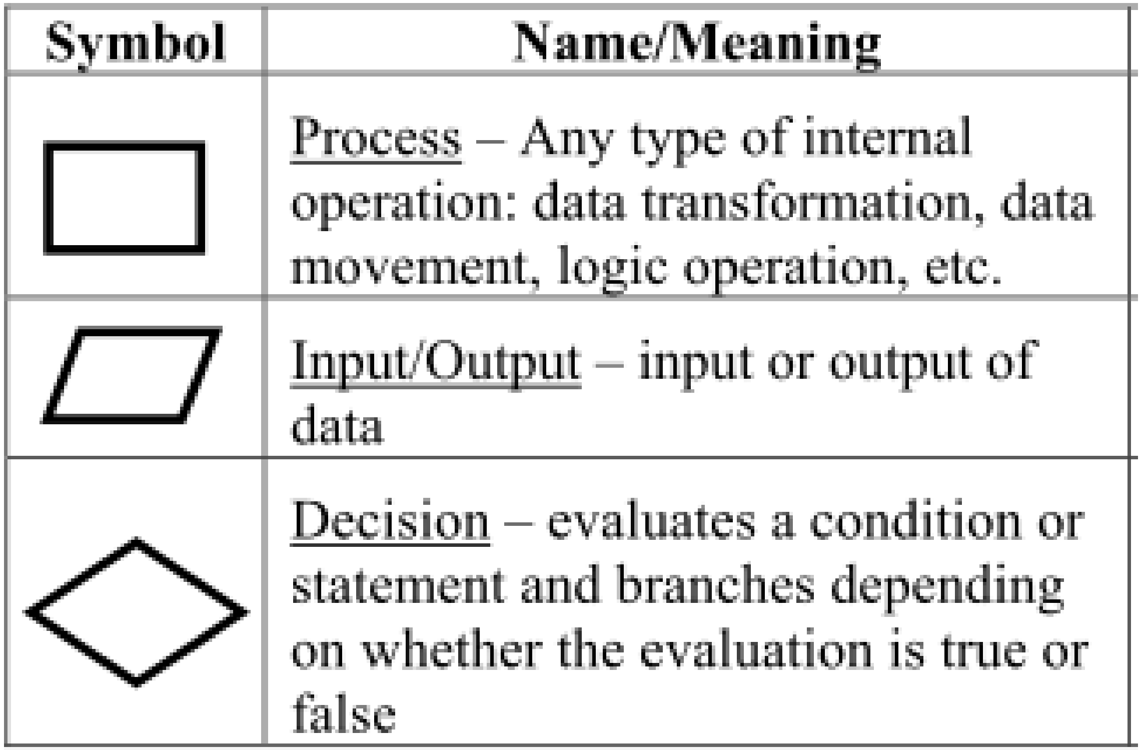


Pseudocode Example

Flowchart

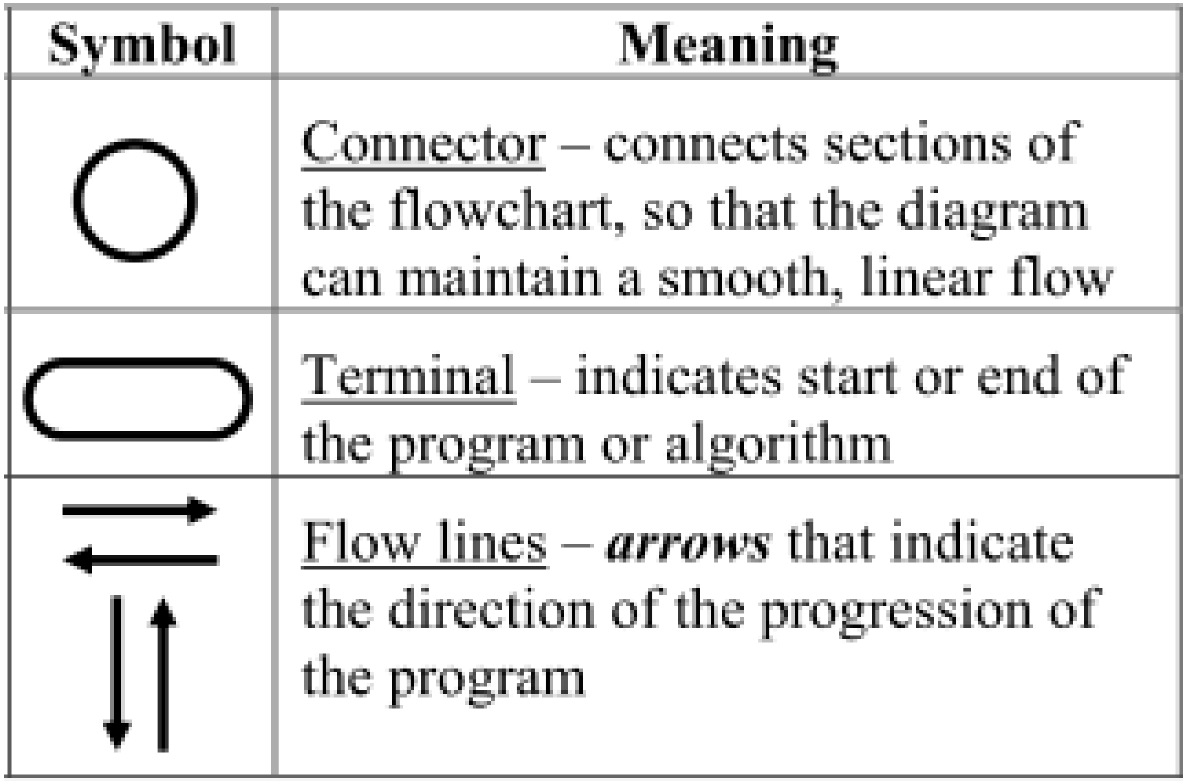
- Tools used to design programs
- Diagrams which graphically depicts the steps
- Three types of symbols in flowchart: ovals, parallelograms and rectangle
- Symbols are connected by arrows that represent the flow of program

Flowchart - Symbol



Symbol	Name/Meaning
□	Process – Any type of internal operation: data transformation, data movement, logic operation, etc.
▱	Input/Output – input or output of data
◇	Decision – evaluates a condition or statement and branches depending on whether the evaluation is true or false

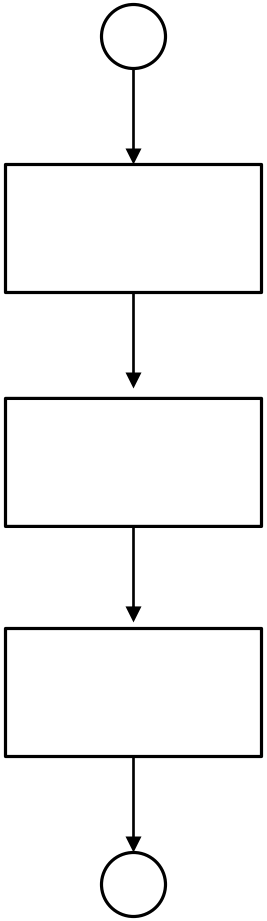
Flowchart - Symbol



Symbol	Meaning
○	Connector – connects sections of the flowchart, so that the diagram can maintain a smooth, linear flow
⬭	Terminal – indicates start or end of the program or algorithm
→ ← ↓ ↑	Flow lines – ***arrows*** that indicate the direction of the progression of the program

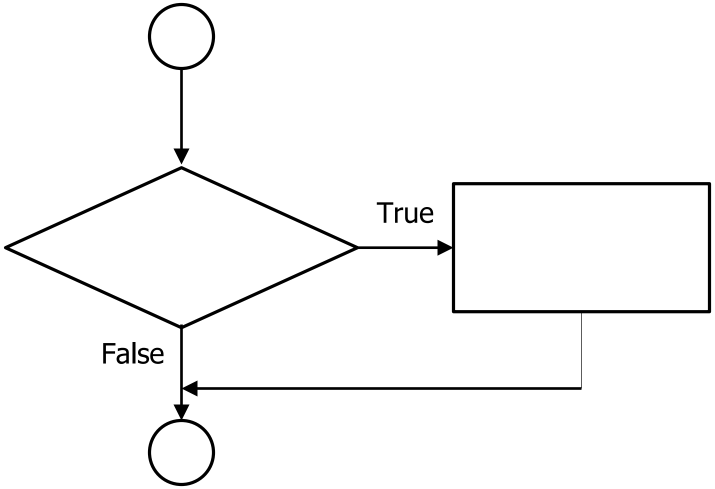
Flowchart Constructs

Sequence



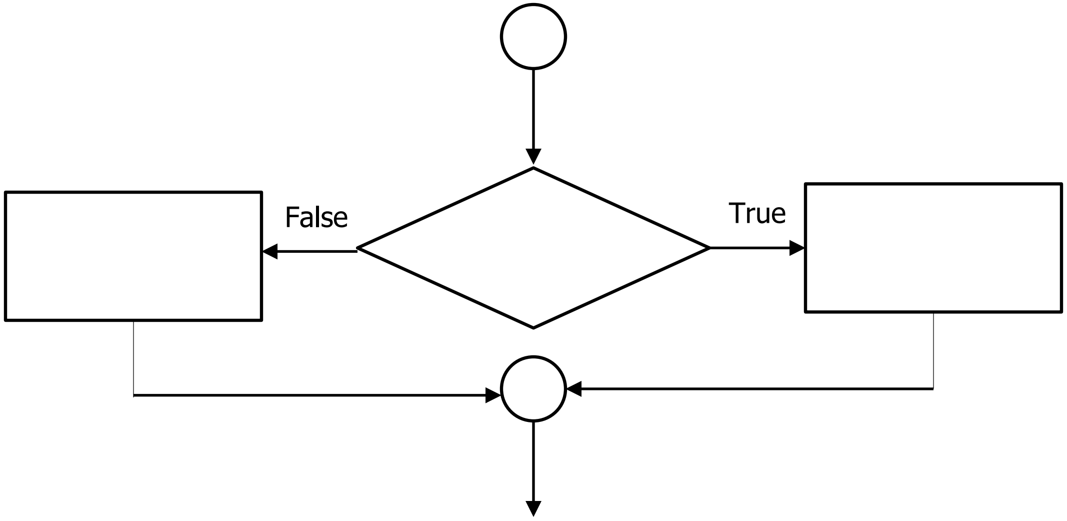
Flowchart Constructs

Single selection: IF



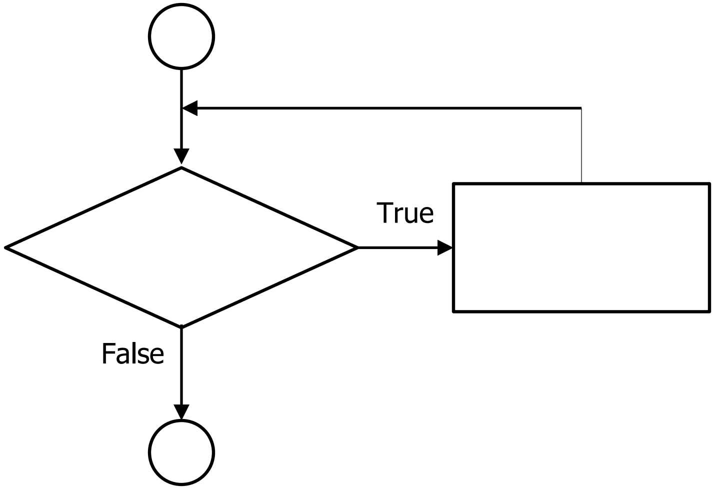
Flowchart Constructs

Double selection: IF… ELSE



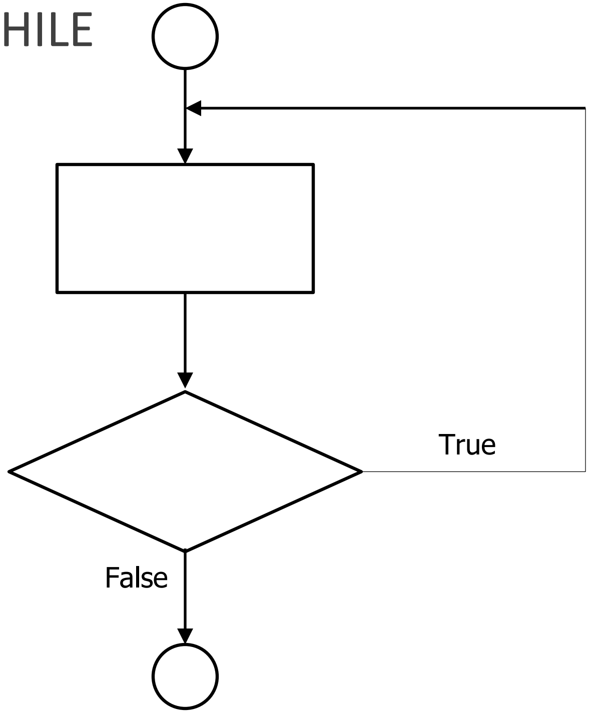
Flowchart Constructs

selection: WHILE



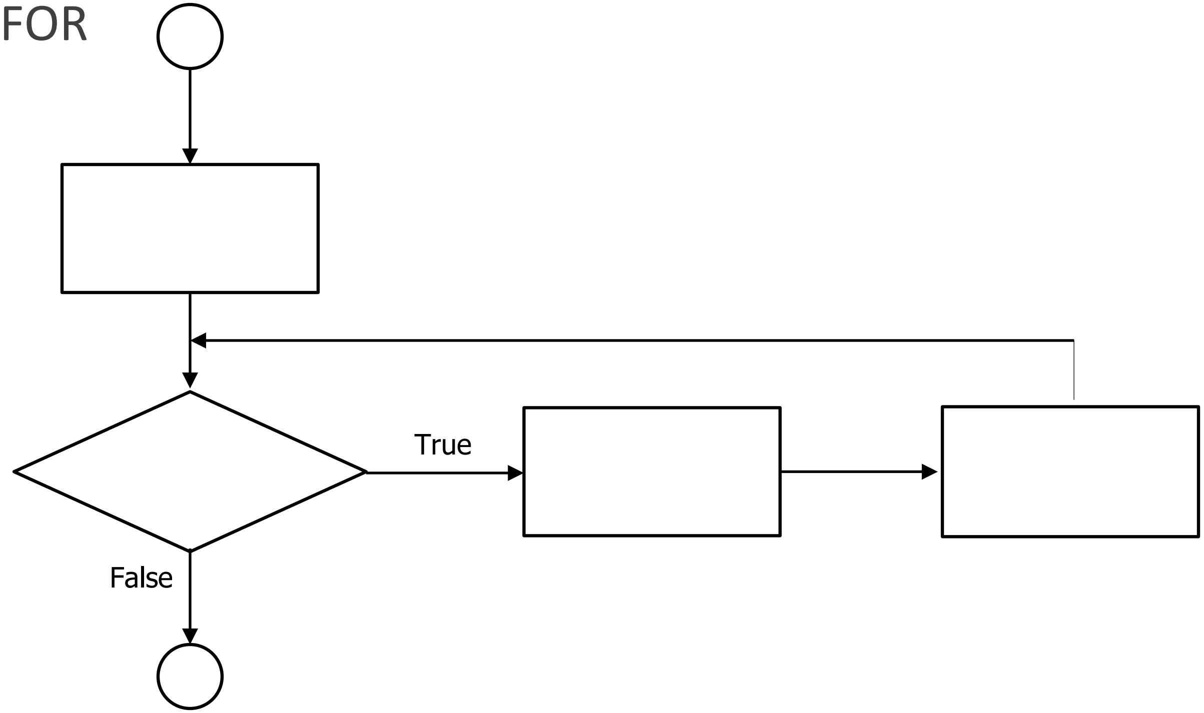
Flowchart Constructs

selection: DO…WHILE

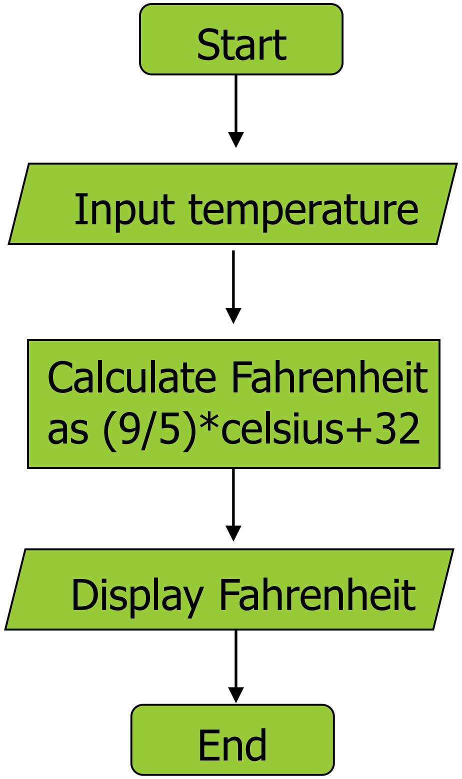


Flowchart Constructs

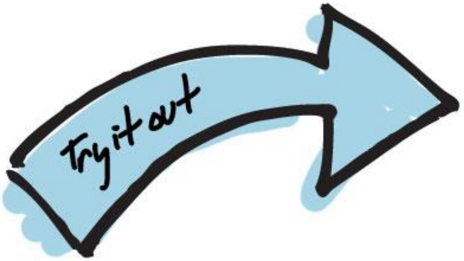
Repetition: FOR



Flowchart: Temperature Converter Program



Flowchart Example



Programming requirements:

Express an algorithm to get two numbers from the user(dividend and divisor), testing to make sure that the divisor number is not zero, if it is, ask user to input again otherwise display their quotient

Flowchart Example

Example Program: Temperature Converter

Implementation: Convert the design in the form of pseudocode or flow chart to computer program using programming language

Example Program: Temperature Converter

```
#convert.py
# A program to convert Celsius temps to Fahrenheit
# by: Susan Computewell

def main():
    celsius = eval(input("What is the Celsius temperature? "))
    fahrenheit = (9/5) * celsius + 32
    print("The temperature is ",fahrenheit," degrees Fahrenheit.")

main()
```

Example Program: Temperature Converter

Test/Debug: Test the written program

```
>>>
What is the Celsius temperature? 0
The temperature is  32.0  degrees Fahrenheit.
>>> main()
What is the Celsius temperature? 100
The temperature is  212.0  degrees Fahrenheit.
>>> main()
What is the Celsius temperature? -40
The temperature is  -40.0  degrees Fahrenheit.
>>>
```

Example Program: Temperature Converter

Maintenance – if the program is used, the programmer should perform regular maintenance to ensure the accuracy of the result

Elements of Programs

Names

- Names are given to variables (celsius, fahrenheit), modules (main, convert), etc.
- These names are called *identifiers*
- Every identifier must begin with a letter or underscore (“_”), followed by any sequence of letters, digits, or underscores.
- Identifiers are case sensitive.

Elements of Program

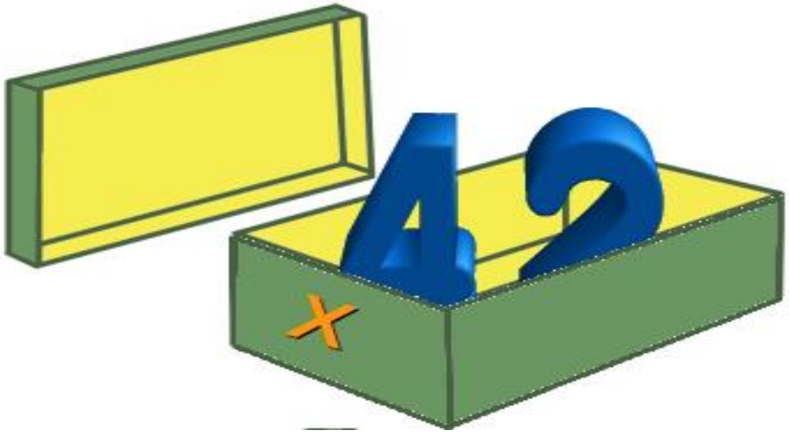
- These are all different, valid names
  - X
  - Celsius
  - Spam
  - spam
  - spAm
  - Spam_and_Eggs
  - Spam_And_Eggs

Elements of Program

To represent something

Different entities: a value, a program, a set of data or a file

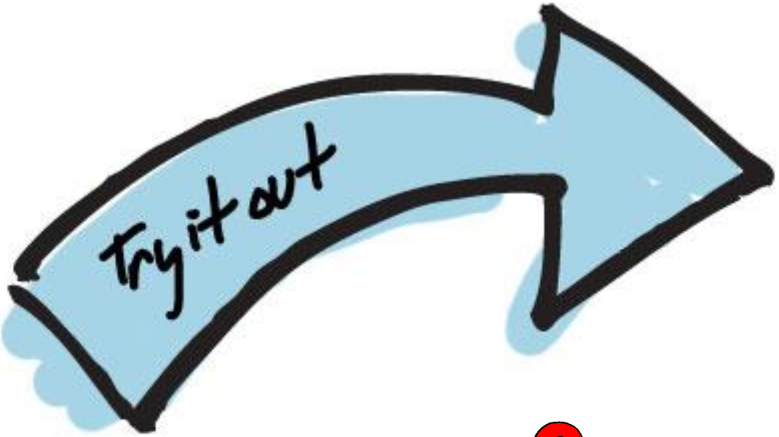
Variable – store an object



Have meaningful names

To improve readability
- Must begin with a letter or underscore (_)
- May contain combination of letters, numbers and underscore
- Any length
- Case sensitive

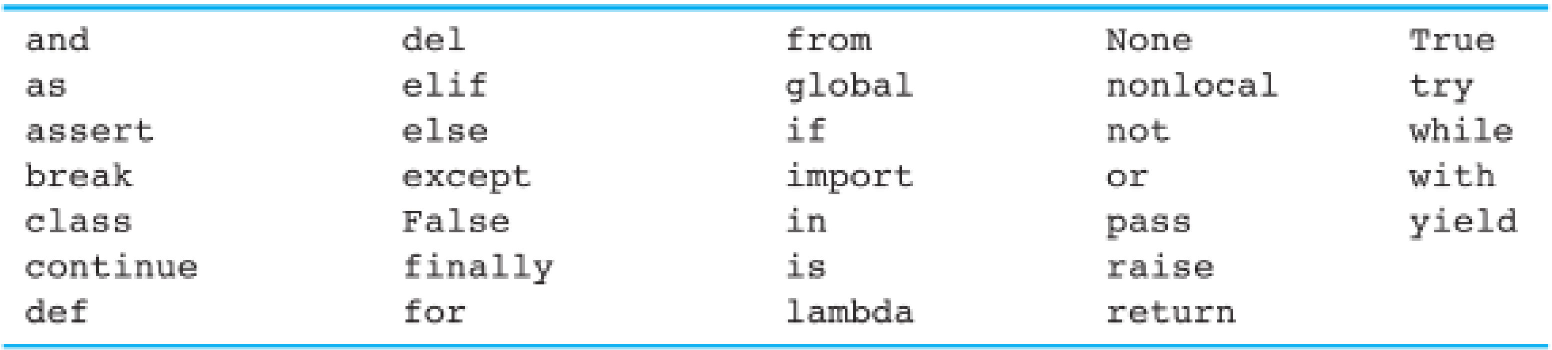
Variable Names



1. Xyzzxxyx
2. xyz
3. burger&lobster
4. burger_lobster
5. seven11
6. 7/11

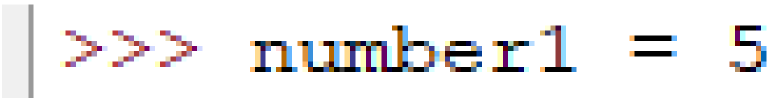
Elements of Program

- Some identifiers are part of Python itself. These identifiers are known as *reserved words*. This means they are not available for you to use as a name for a variable in your program.



and	del	from	None	True
as	elif	global	nonlocal	try
assert	else	if	not	while
break	except	import	or	with
class	False	in	pass	yield
continue	finally	is	raise	
def	for	lambda	return	

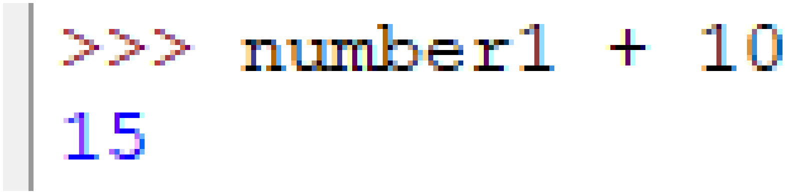
Elements of Program– Statements and Expressions



```
>>> number1 = 5
```

Statement

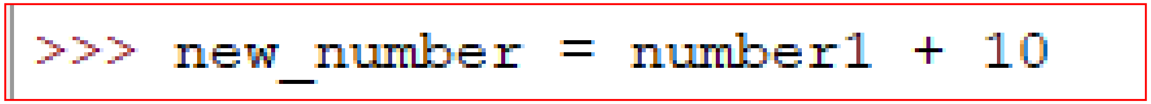
- Perform task
- Does not return value



```
>>> number1 + 10
15
```

Expression

- Combination of values and operations
- Create a new value and return it

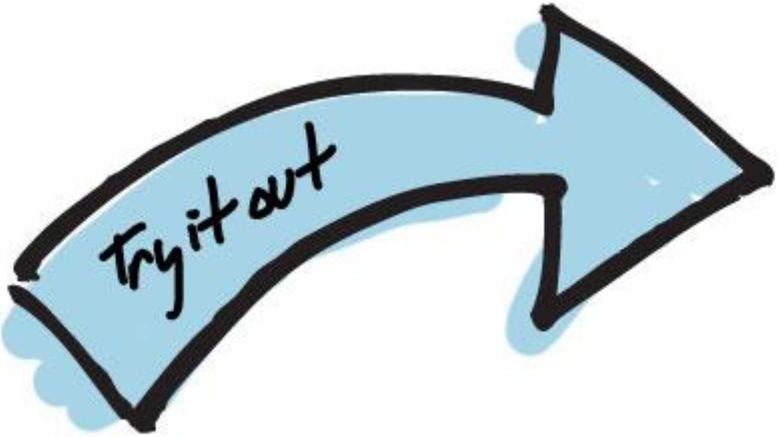


```
>>> new_number = number1 + 10
```

Statement **OR** Expression

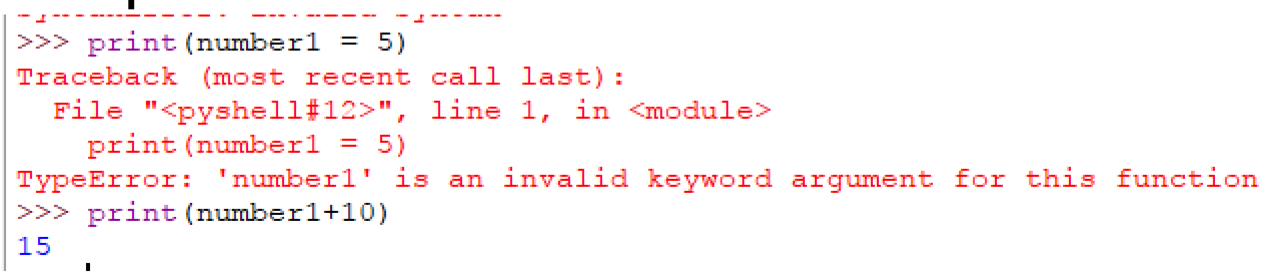
Note 1

You can print an expression but not a statement



```
>>> print (number1 = 5)
>>> print (number 1 + 10)
```

Output:



```
>>> print(number1 = 5)
Traceback (most recent call last):
  File "<pyshell#12>", line 1, in <module>
    print(number1 = 5)
TypeError: 'number1' is an invalid keyword argument for this function
>>> print(number1+10)
15
```

Elements of Program - NameError

```
>>> x = 5
>>> x
5
>>> print(x)
5
>>> print(spam)

Traceback (most recent call last):
  File "<pyshell#15>", line 1, in -toplevel-
    print spam
NameError: name 'spam' is not defined
>>>
```

NameError is the error when you try to use a variable without a value assigned to it.

Elements of Program - Operators

- Simpler expressions can be combined using *operators*.
- +, -, *, /, **
- Spaces are irrelevant within an expression.
- The normal mathematical precedence applies.
- ((x1 – x2) / 2*n) + (spam / k**3)

Elements of Program

Output Statements

- A print statement can print any number of expressions.
- Successive print statements will display on separate lines.
- A bare print will print a blank line.

Elements of Program

```
print(3+4)
print(3, 4, 3+4)
print()
print(3 + 4)
print("The answer is", 3+4)
```

Output

```
7
3 4 7

7
The answer is 7
```

Elements of Program - Statements

Simple Assignment

<variable> = <expr>
variable is an identifier, expr is an expression

The expression on the RHS is evaluated to produce a value which is then associated with the variable named on the LHS.

Assignment Statements

```
x = 3.9 * x * (1-x)

fahrenheit = 9/5 * celsius + 32

x = 5
```

Assignment Statements

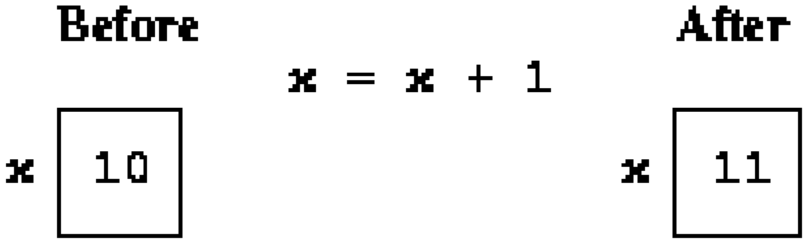
Variables can be reassigned as many times as you want!

```
>>> myVar = 0
>>> myVar
0
>>> myVar = 7
>>> myVar
7
>>> myVar = myVar + 1
>>> myVar
8
>>>
```

Assignment Statements

Variables are like a box we can put values in.

When a variable changes, the old value is erased and a new one is written in.

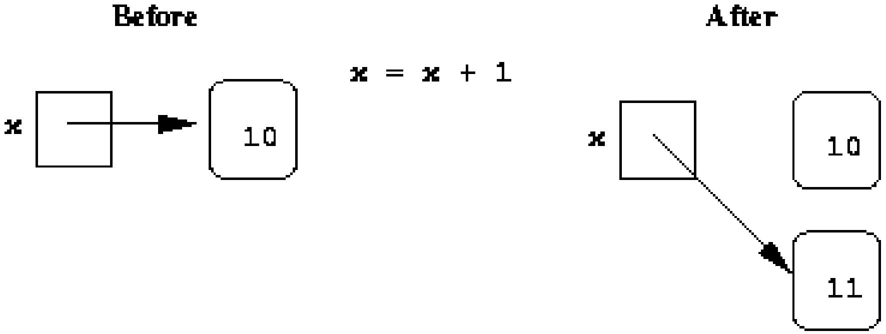


Assignment Statements

Technically, this model of assignment is simplistic for Python.

Python doesn't overwrite these memory locations (boxes).

Assigning a variable is more like putting a “sticky note” on a value and saying, “this is x”.



Assigning Input

The purpose of an input statement is to get input from the user and store it into a variable.

```
<variable> = eval(input(<prompt>))
```

Assigning Input

First the prompt is printed

The `input` part waits for the user to enter a value and press <enter>

The expression that was entered is `eval`uated to turn it from a string of characters into a Python value (a number).

The value is assigned to the variable.

Input and Output example

Input

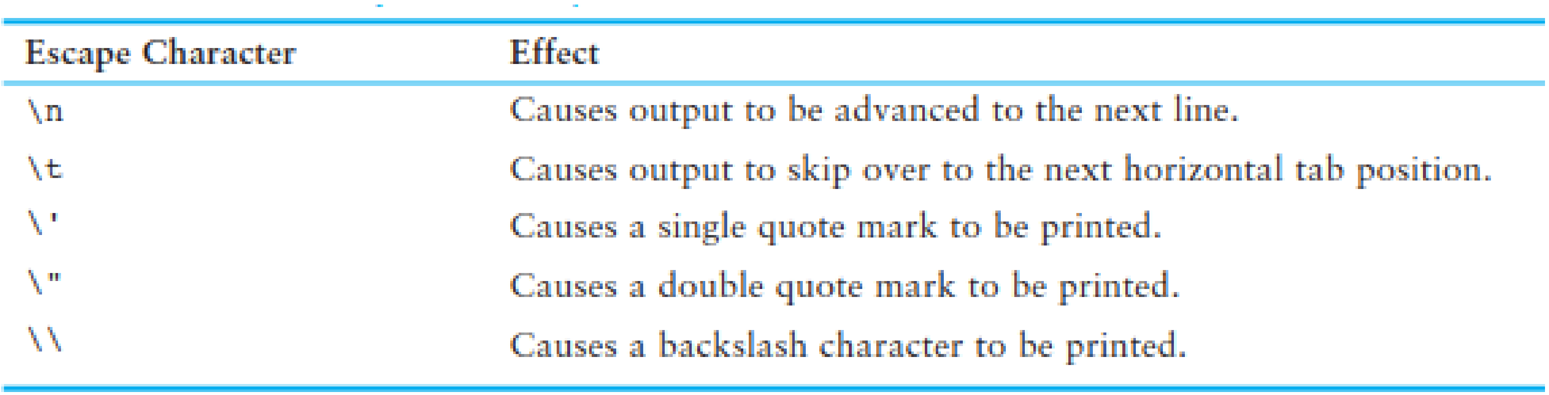
```
name = eval(input (“What is your name?”))

age = eval(input(“What is your name?”))
```

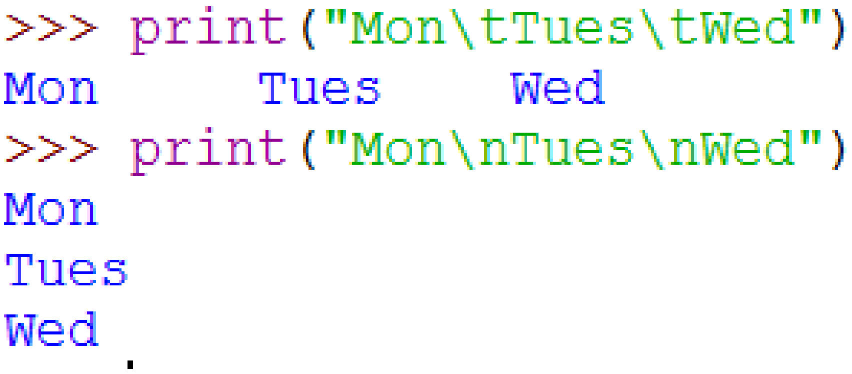
Output

```
print(“Name:”,name)
print(“Age:”, age)
```

Output – escape characters



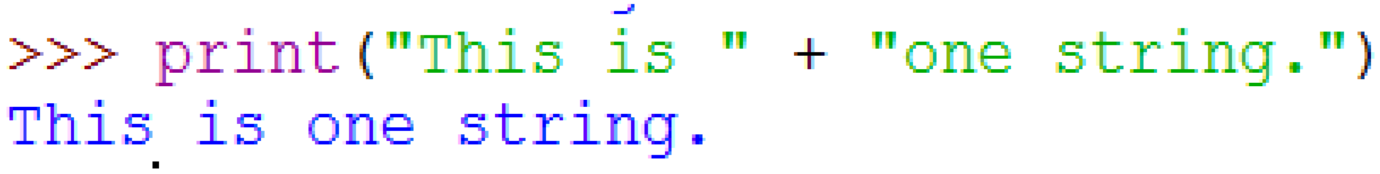
Escape Character	Effect
\n	Causes output to be advanced to the next line.
\t	Causes output to skip over to the next horizontal tab position.
\'	Causes a single quote mark to be printed.
\"	Causes a double quote mark to be printed.
\\	Causes a backslash character to be printed.



```
>>> print("Mon\tTues\tWed")
Mon     Tues    Wed
>>> print("Mon\nTues\nWed")
Mon
Tues
Wed
```

Output – Displaying multiple items

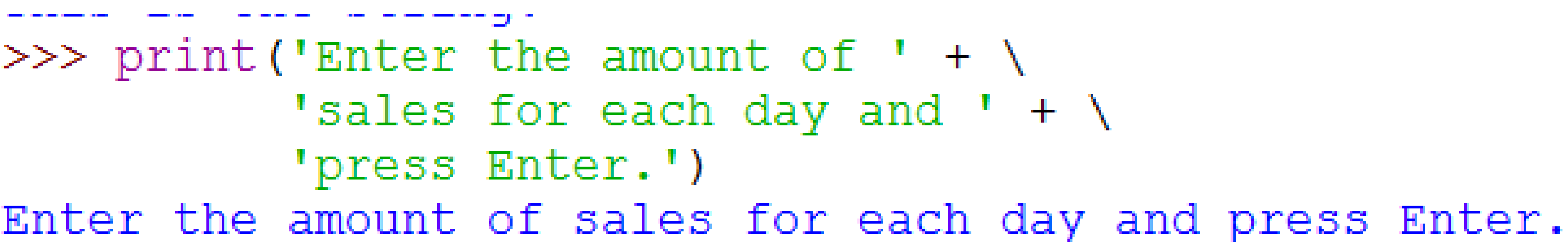
Using ‘+’ operator to perform string concatenation



```
>>> print("This is " + "one string.")
This is one string.
```

Output – Displaying multiple items

If the command is too length, break them into multiple lines using ‘\’



```
>>> print('Enter the amount of ' + \
          'sales for each day and ' + \
          'press Enter.')
Enter the amount of sales for each day and press Enter.
```

Simultaneous Assignment

Several values can be calculated at the same time

<var>, <var>, … = <expr>, <expr>, …

Evaluate the expressions in the RHS and assign them to the variables on the LHS

Simultaneous Assignment

```
sum, diff = x+y, x-y
```

How could you use this to swap the values for x and y?

- Why doesn’t this work?

  ```
  x = y
  y = x
  ```

We could use a temporary variable…

Simultaneous Assignment

We can swap the values of two variables quite easily in Python!

- x, y = y, x

```
>>> x = 3
>>> y = 4
>>> print x, y
3 4
>>> x, y = y, x
>>> print x, y
4 3
```

Simultaneous Assignment

We can use this same idea to input multiple variables from a single input statement!

Use commas to separate the inputs

```
def spamneggs():
   spam, eggs = eval(input("Enter # of slices of spam followed by # of eggs: "))
   print ("You ordered", eggs, "eggs and", spam, "slices of spam. Yum!“)
```

```
>>> spamneggs()
Enter the number of slices of spam followed by the number of eggs: 3, 2
You ordered 2 eggs and 3 slices of spam. Yum!
>>>
```

Exercise

Develop a program to calculate the BMI given weight in kilogram and height in cm

BMI = weight(kg) / height(m) X height(m)

References

Gaddis, T. (2018). *Starting out with Python* (4th ed.). Essex, England: Pearson Education Limited.