INT 489 Selected Topics IN IT

Dr. Khaled Mostafa Reda

Sinai University (SU)

Faculty of Information Technology and Computer Science (FIT)

E-Mail: khaled.Mostafa@su.edu.eg

Welcome

To the Selected Topics IN IT Course

Code: INT 489

Fall Term 2023

TODAY

- Course information.
- Course Policy.
- What is computation?
- Python basics.
- Mathematical operations.
- Python variables and types
- **NOTE: slides and code files up each lecture**
  - Highly encourage you to download it before the lecture
  - Take notes and run code files when I do
  - Bring computers to answer **in-class practice exercises!**

Welcome to Selected Topics IN IT Course

- **Important Course Information**

Group	Day	Hours	Locations
A	Sunday	14:40-1620:00	B2013

Course Policy

- **Grading:**
  - 5% Course Work (CW)
  - 10% Oral/Practical

  } **15%**
  - 25% on one Term Exam (T.E).
  - 60% on the Final Exam (F.E).
  - **100% Total Mark**
  - **TALKING** and **SLEEPING** are strongly forbidden during class.
  - **Late assignments**
  - **Plagiarism**

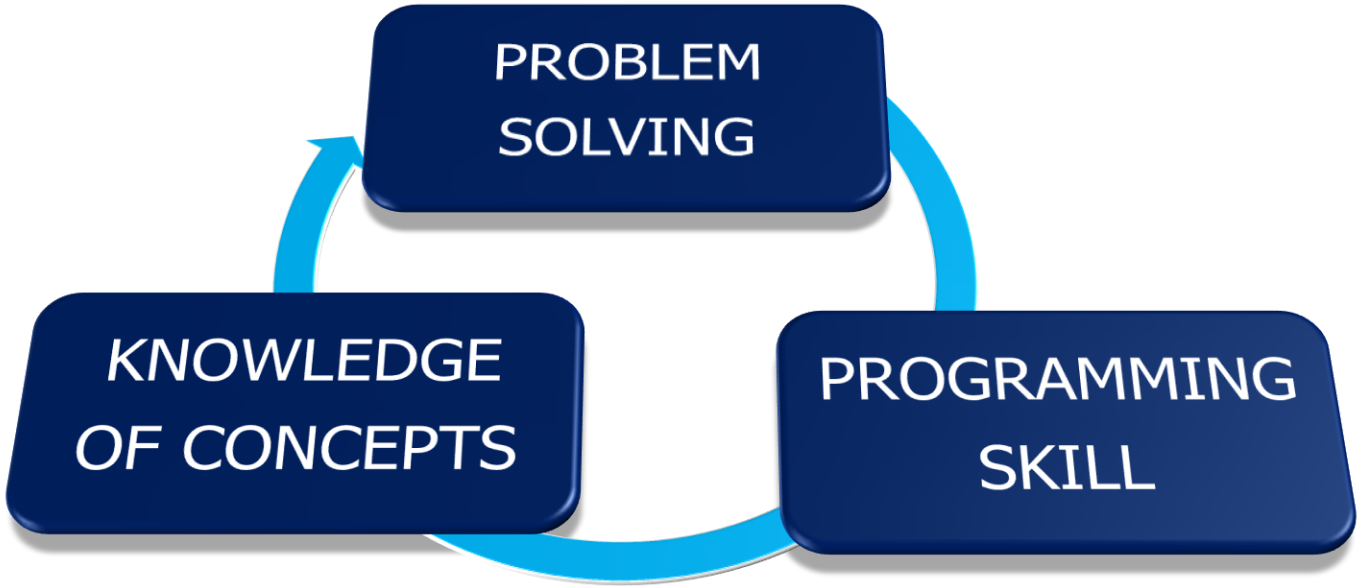
Recitations

- Not mandatory
- Two recommendation
  1) Lecture review: **review** lecture material
     - if you missed the lecture
     - if you need a different take on the same concepts
  2) Problem-solving teaches you **how to solve** programming problems
     - Useful if you don't know how to set up pseudocode from summarized words
     - walk you through how to approach solving the problem
     - brainstorm code solutions along with the recitation instructor
     - Will post solutions after

Fast Paced Course

- **Position yourself to succeed!**
  - Read the summaries as it comes out and come back to them later.
  - use late days in emergency situations
- **New to programming? Practice. Practice? Practice!**
  - Download the code and the lecture and follow along
  - Don’t be afraid to try out Python commands!

Practice



Program development cycle

- **Program?**
  - a collection of instructions, that directs the computer hardware to accomplish a certain task.
- **Performing A Task On The Computer**
  - Determine what the **output** should be; that is, exactly what the task should produce.
  - Identify the data, or **input**, necessary to obtain the output.
  - The last step is to determine how to **process** the input.
  - Determine what formulas or ways of doing things can be used to obtain the output.

Program development cycle Cont.,

- **PROGRAM PLANNING**
  - Many programmers plan their programs using a sequence of steps.
- **Program development cycle**.
  - **Analyze:** Define the problem.
  - **Design:** Plan the solution to the problem. Find a logical sequence of precise steps that solve the problem. Such a sequence of steps is called an **algorithm**.
  - **Code:** Translate the algorithm into a programming language.
  - **Test and debug:** Locate and remove any errors in the program.
  - **Complete the documentation:** Organize all the material that describes the program.

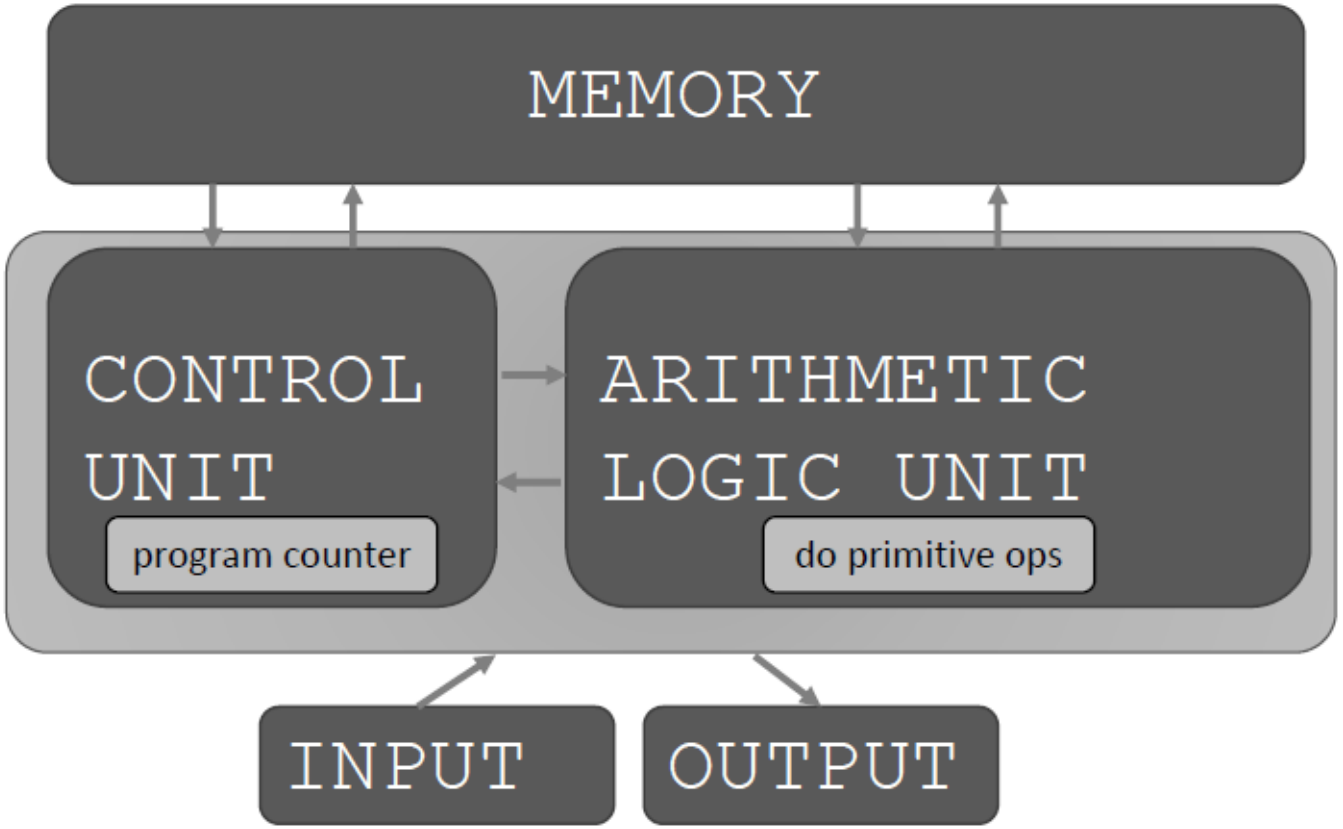
What Does A Computer Do

- Fundamentally:
  - performs **calculations**
    - a billion calculations per second!
  - remembers results
    - 100s of gigabytes of storage!
- What kinds of calculations?
  - built-in to the language
  - ones that you define as the programmer
- Computers only know what you tell them

Computers are Machines

- How to capture a recipe in an automatic process
- **Fixed program** computer
  - Calculator
- **Stored program** computer
  - Machine stores and executes instructions

Basic Machine Architecture



Stored Program Computer

- Sequence of **instructions stored** inside the computer
  - built from a predefined set of primitive instructions
    1) arithmetic and logic
    2) simple tests
    3) moving data
- A special program (interpreter) **executes each instruction in order**
  - use tests to change the flow of control through a sequence
  - stop when done

What is a recipe

1. Sequence of simple **steps**
2. **flow of control** process that specifies when each step is executed
3. a means of determining **when to stop**

1+2+3 = an **algorithm**!

Algorithms

- Any solvable computing problem can be solved by the execution of a series of actions in a specific order. A procedure for solving a problem in terms of the actions to execute and the order in which these actions execute is called an algorithm.

Programming Fundamentals

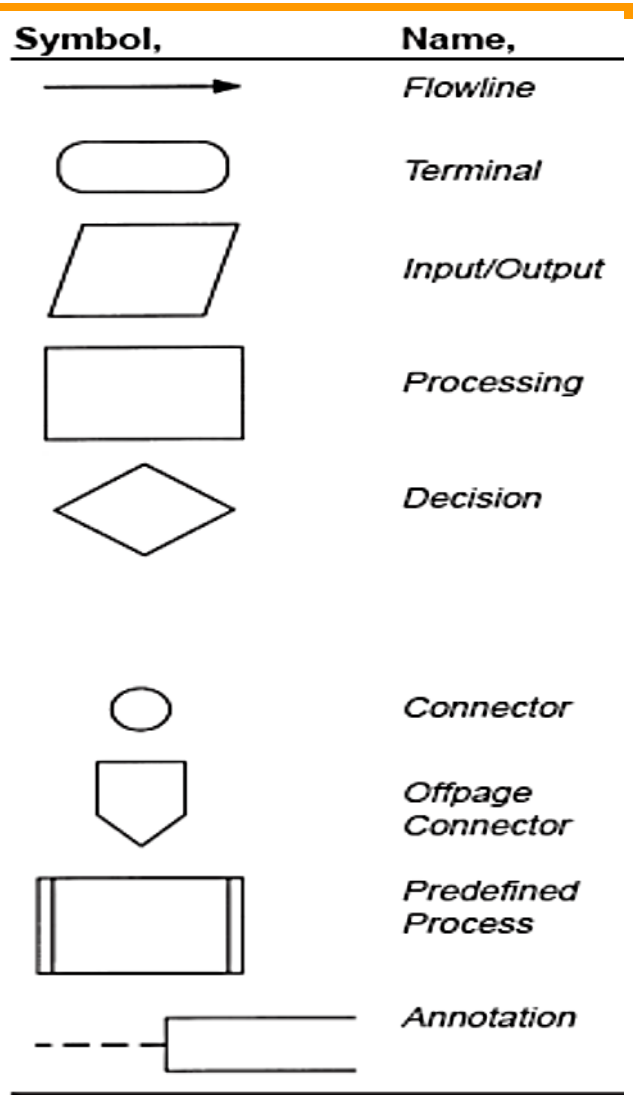
- An algorithm is a special type of procedure.
- An algorithm is a procedure that has the following characteristics;
  1. terminates,
  2. is effective (i.e., executable), and
  3. is unambiguous.

Programming Fundamentals

- To describe an algorithm, we need to specify Four things:
  1. The input(s) to the algorithm
  2. Operations, mathematical formulas, and steps of the algorithm,
  3. how these steps are to be structured to achieve the required objectives, and
  4. a language in which we can unambiguously describe the steps and structure of the algorithm

Programming Tools

- Flowcharts, a Graphical representation that depicts the logical steps to carry out a task and show how the steps relate to each other.
- Pseudo-code, Uses English-like phrases with some Python terms to outline the task.



Example 1

- Problem: To compute and display the square of a given number.
- Analysis:

input: any number x; **output:** y=square of x;
Relation: y=x*x

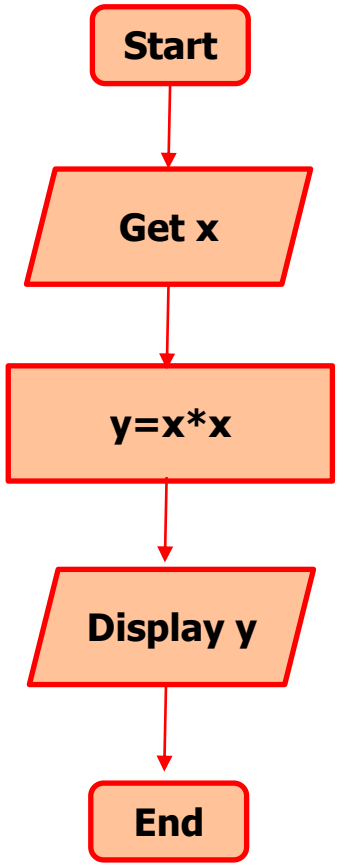
Design

1- Pseudo-code

1. Ask the user to enter the number
2. Store it in variable (memory location) *x*
3. Set *y to x * x*
4. Print *y*

Sequential Structure

2- Flowcharts



Example 2

- Problem: To compute the absolute value of a given number.
- Analysis:

Input; any number x; output y=absolute value (x);

Relation: y= x if x≥ 0

Design = -x if x<0

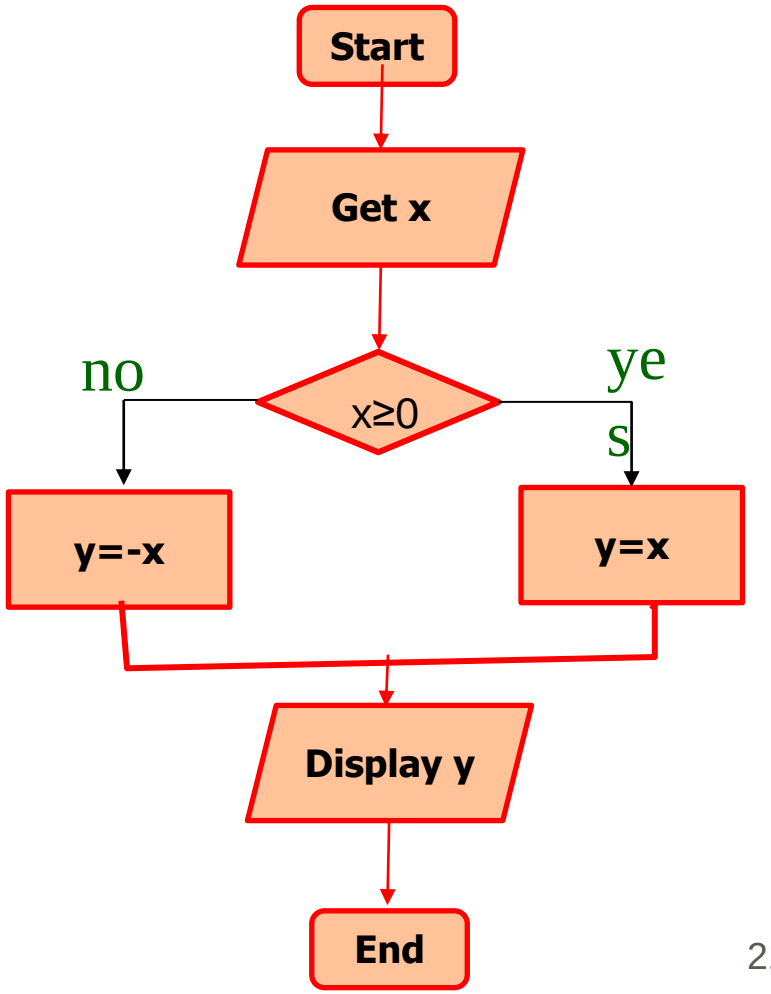
1- Pseudo-code

1. Ask the user to enter a number
2. get the number and store it in variable x
3. if x ≥ 0 then
4. set y to x
5. else
6. set y to –x
7. Display y

Sequential Structure

Conditional Structure

2- Flowcharts



Python Programs

- A **program** is a sequence of definitions and commands
  - definitions **evaluated**
  - commands **executed** by a Python interpreter in a shell
- **Commands** (statements) instruct the interpreter to do something.
- Can be typed directly in a **shell** or stored in a **file** that is read into the shell and evaluated

Where to get Python

- You can download your own copy, it is free
  - Download Python recommended
  - Download Spyder
- More advanced tasks (data science, machine learning)
  - Download Anaconda
  - Not recommended at this level
- You can use online compilers/IDEs
  - Programiz recommended

Python IDE

- What is an IDE?
  - An Integrated Development Environment allows you to write programs in a programming language and provides extra tools to help the process
    - Syntax highlighting
    - Code completion
    - Bracket completion/matching
    - Debugging
    - Profiling
- PyCharm is a very powerful and popular IDE for Python
- IDLE comes built-in with any Python distribution
- Eric Python IDE , Spyder, Eclipse PyDev
- Sublime Text, VS Code, Atom, etc. are generic editors that can be extended with Python capabilities

Python Indentation

- Indentation refers to the spaces at the beginning of a code line.
- In other programming languages the indentation in code is for readability only, the indentation in Python is very important.
- Python uses indentation to indicate a block of code.
-
```
if 5 > 2:
  print("Five is greater than two!")
```
- Python will give you an error if you skip the indentation:

```
if 5 > 2:
print("Five is greater than two!")
```

Python Indentation, cont.

- The number of spaces is up to you as a programmer, but it has to be at least one.

```
if 5 > 2:
 print("Five is greater than two!")
if 5 > 2:
    print("Five is greater than two!")
```

- You have to use the same number of spaces in the same block of code, otherwise, Python will give you an error:

```
if 5 > 2:
 print("Five is greater than two!")
    print("Five is greater than two!")
```

Objects

- Programs manipulate **data objects**
- Objects have a **type** that defines the kinds of things programs can do to them
  - Ali is a human so he can walk, speak English, etc.
- objects are
  - scalar (cannot be subdivided)
  - non-scalar (have an internal structure that can be accessed)

Scalar Objects

- int – represent **Integer**, ex. 5
- float – represent **real numbers**, ex. 3.27
- bool – represent **Boolean** values True and False
- NoneType – Special and has one value, None
- can use type() to see the type of an object
- >>> type(5)
  - int
- >>> type(3.0)
  - float

Non-Scalar Objects

- Strings are non-scalar objects. They have internal structures.

```
name = “Hasen"
print(name[2:4])
## se
```

- We can construct new non-scalar objects.
- Object-oriented programming is the art of programming using non-scalar objects.

Types Conversions (CAST)

- We can convert objects of one type to another
  - Not all types are convertible!
- float(3) converts integer 3 to float 3.0
- int(3.9) truncates float 3.9 to integer 3
- float(3)
- ## 3.0
- int(3.9)
- ## 3
- int(“Hasen")
- ## ValueError: invalid literal for int() with base 10: ‘Hasen'

Printing To Console

- To show output from code to a user, use the print command

```
print("Hello World!")
## Hello World!
print(3 + 2)
## 5
print("My age is", 41)
## My age is 41
```

Expressions

- Combine objects and operators to form expressions
- An expression has a value, which has a type
- Syntax for a simple expression
- <object> <operator> <object>

Operators On ints and floats

- i+j the sum
- i-j the difference
- i*j the product
- i/j the division
  - For the sum, the difference, and the product, if both objects are integers, then the result is an integer.
  - If one or both are floats, then the result is a float.
  - For the division, the result is always a float.
- i%j the remainder when i is divided by j
- i**j i to the power of j

Operators on ints and floats

- 3 + 5
  - ## 8
- 3 - 5
  - ## -2
- 3 * 5
  - ## 15
- 3 / 5
  - ## 0.6
- 32 % 5
  - ## 2
- 3 ** 4
  - ## 81

Simple Operations

- parentheses are used to tell Python to prioritize operations
  - 3 * (2 + 5)
  - ## 21
- operator precedence without parentheses
  - **
  - *
  - /
  - + and – executed left to right, as appear in the expression
- 3 * 2 + 5
  - ## 11

Binding Variables And Values

- Equal sign is an assignment of a value to a variable name
- # variable = value
- pi = 3.14159
- pi_approx = 22/7
- value stored in computer memory
- an assignment binds a variable name to a value
- retrieve the value associated with the variable name by invoking the name, by typing pi
- pi
- ## 3.14159
- pi_approx
- ## 3.142857142857143

Multiple Assignments

- you can assign multiple values to variables at once
- a, b = 3, 5
- Print(a)
- ## 3
- Print(b)
- ## 5

Variable Naming

- it is important to use clear and understandable names for variables
- also, using parenthesis in expressions helps with code readability
- a, b, c = 5, 8, 3.14
- d = c * a ** 2 * b

- r, h, pi = 5, 8, 3.14
- V_cyl = pi * (r ** 2) * h
- Vcyl = π* r^2*h

Variables Cont.,

- **Single or Double Quotes?**
  - String variables can be declared either by using single or double quotes:

```
x = "John"
# is the same as
x = 'John’
```

- **Case-Sensitive**
  - Variable names are case-sensitive.

```
a = 4
A = "Sally"
#A will not overwrite a
```

Comments

- You can enrich your code by adding comments
- to add a comment, start a line with #
- lines starting with # are ignored by Python
- Python has the commenting capability for the purpose of in-code documentation.
- Comments can be used to explain Python code.
- Comments can be used to make the code more readable.
- # Radius, height, and pi are defined
- r, h, pi = 5, 8, 3.14
- # The volume of the cylinder is computed
- # Volume is equal to height times the base area
- V_cyl = pi * (r ** 2) * h

Comments cont.

- Multi-Line Comments
- Python does not really have a syntax for multi-line comments.
- To add a multiline comment, you could insert a # for each line:
-

```
#This is a comment written in
#more than just one line
print("Hello, World!")
```

- Since Python will ignore string literals that are not assigned to a variable, you can add a multiline string (triple quotes) in your code, and place your comment inside it:
-

```
"""
```

-

```
This is a comment written in
more than just one line
"""
print("Hello, World!")
```

Abstracting Expressions

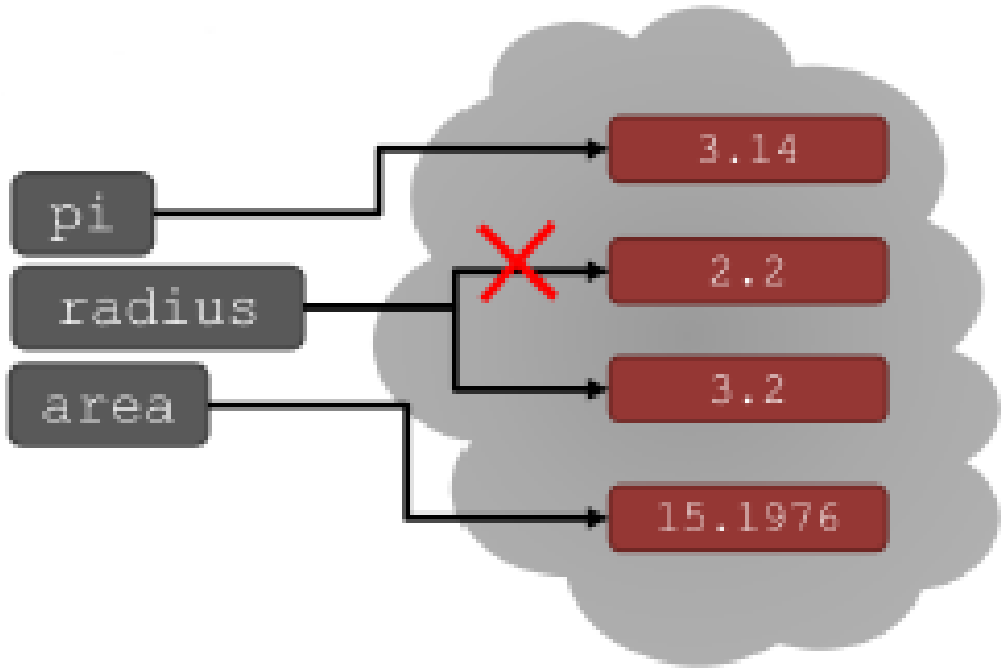
- Why give names to values of expressions?
- to reuse names instead of values
- easier to change code later
- pi = 3.14159
- radius = 2.2
- area = pi * (radius ** 2)
- print(area)
- ## 15.205295600000001

Programming vs MATH

- in programming, variables do not get automatically updated
- pi = 3.14159
- radius = 2.2
- # area of a circle
- area = pi * (radius**2)
- print(area)
- ## 15.205295600000001
- radius = radius + 1
- print(radius)
- ## 3.2
- print(area)
- ## 15.205295600000001
- area = pi * (radius**2)
- print(area)
- ## 32.169881600000004

Changing Bindings

- can re-bind variable names use new assignment statements
- previous value may still be stored in memory but lost the handle for it
- value for the area does not change until you tell the computer to do the calculation again
- pi = 3.14
- radius = 2.2
- area = pi * (radius**2)
- radius = radius + 1



Assignment

1. Analyze and design an algorithm to calculate the area of a rectangle, by drawing flowcharts and writing pseudocode. Finally, write a Python program to express your design.

2. Analyze and design an algorithm to calculate the area and the circumference of a circle by drawing the flowchart and writing pseudocode to solve this problem, where the area of the circle (A) = π * r^2, the circumference of a circle (C)=2*π*r, r…radius of the circle, and π (pi) = 3.14. Finally, write a Python program to express your design.

3. Analyze and design an algorithm to find the roots of quadratic equations $ax^2 + bx + c = 0$ and print a message the roots of the equation are r1 and r2 by drawing the flowchart and writing pseudocode according to the following:

$$D = b^2 - 4ac \qquad r1 = \frac{-b + \sqrt{D}}{2 * a} \qquad r2 = \frac{-b - \sqrt{D}}{2 * a}$$

Finally, write a Python program to express your design.

Thank You