CYBR 520

Lecture 1.B. : Coding in python

West Virginia University
JOHN CHAMBERS COLLEGE OF
BUSINESS AND ECONOMICS

Pseudo code

- Plain language description of steps in an algorithm/ program.
- It is highly encouraged to use it before programming.
- Makes programming easier and shows issues in your logic (if there were any).

WestVirginiaUniversity.
JOHN CHAMBERS COLLEGE OF
BUSINESS AND ECONOMICS

Pseudo code: Example

- Let us say we need to write a program that accepts two values from the user then sums those values.
  - What would be the first step?
  - What would be the second step?
  - See next slide

Pseudo code: Example

```
Begin
        define variable 1 and variable 2
        Display a message: ask users for variable 1 value
        Input variable1
        Display a message: ask users for variable 2 value
        Input variable2
        display (variable 1 + variable 2)
End
```

Variables

- The term variables refers to something that can change. Even if you are not familiar with the idea of a variable, if you took algebra or trigonometry in school, you have used variables in the past.
- You would have seen them written similarly to x = 3 + 5 , followed by the request Find the value of x.

Variables

- The easiest way to describe a variable in a computer is to imagine a box. We can take a box and give it a name such as “Frank”.
- We then can put almost anything we want, such as marbles, into “Frank”. Then, instead of asking for the specific contents (i.e., the marbles we put in it), we can ask for “Frank” instead

Working with Variables

- Working with variables in Python is mostly simple; there are more complex variable types seen later will make it more difficult, but these are generally based on simple types.
- We start with a name, such as color, and then assign a value to it using a single equal sign (=).
- When a single equal sign is used in Python, it assigns the value on the right of the equal sign to the variable on the left. For instance, if we were painting a car, we might assign color = "Yellow"

Variable Names

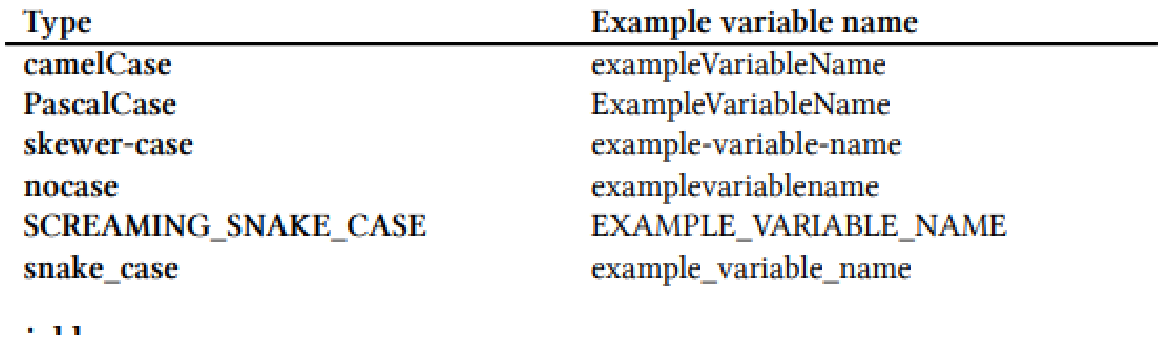
- When working with Python, there several conventions, or suggestions, on how to name variables. These conventions help to encourage the “readability” focus when working with Python. More information regarding conventions can be found in the Python Style Guide .

Variable Names: Make names meaningful

- Variable names should be descriptive of what information, or type of information, they hold.
- For instance, names such as student_id, first_name, and vehicle_type help identify the content of the variable and where it would be used.
- Names such as x, spam, and eggs are not descriptive and can become confusing, especially if reused multiplied times in a script. Names should be lowercase

Variable Names: Names should be lowercase

- Wherever possible, Python variable should be all lower case. Different programmers have preferred variable naming styles such as camelCase, PascalCase, skewer-case, nocase, and even SCREAMING_SNAKE_CASE. Python suggests using what is known as snake_case, all lower-case letters with underscores separating words



Type	Example variable name
camelCase	exampleVariableName
PascalCase	ExampleVariableName
skewer-case	example-variable-name
nocase	examplevariablename
SCREAMING_SNAKE_CASE	EXAMPLE_VARIABLE_NAME
snake_case	example_variable_name

West Virginia University
JOHN CHAMBERS COLLEGE OF
BUSINESS AND ECONOMICS

Variable Types

- Just as there are different types of pets in people's homes, there are different types of variables in Python. It is important to recognize the different types of variables because some types work well together, while other types need special consideration to work together.
- We can see more information about what our variable type is by using the type() built-in Python function.
- We can see an example of this below.
  - String (str) - can be a combination of text, numbers, and spaces, or most anything you can type on the keyboard.
  - Integer (int) - is a simple number that doesn't include a period. Numbers such as 0, 1, 2, -5, and 42 are integers.
  - Float (float) - is a number that includes a period. Numbers such as 3.14 and 1.234 are floats.

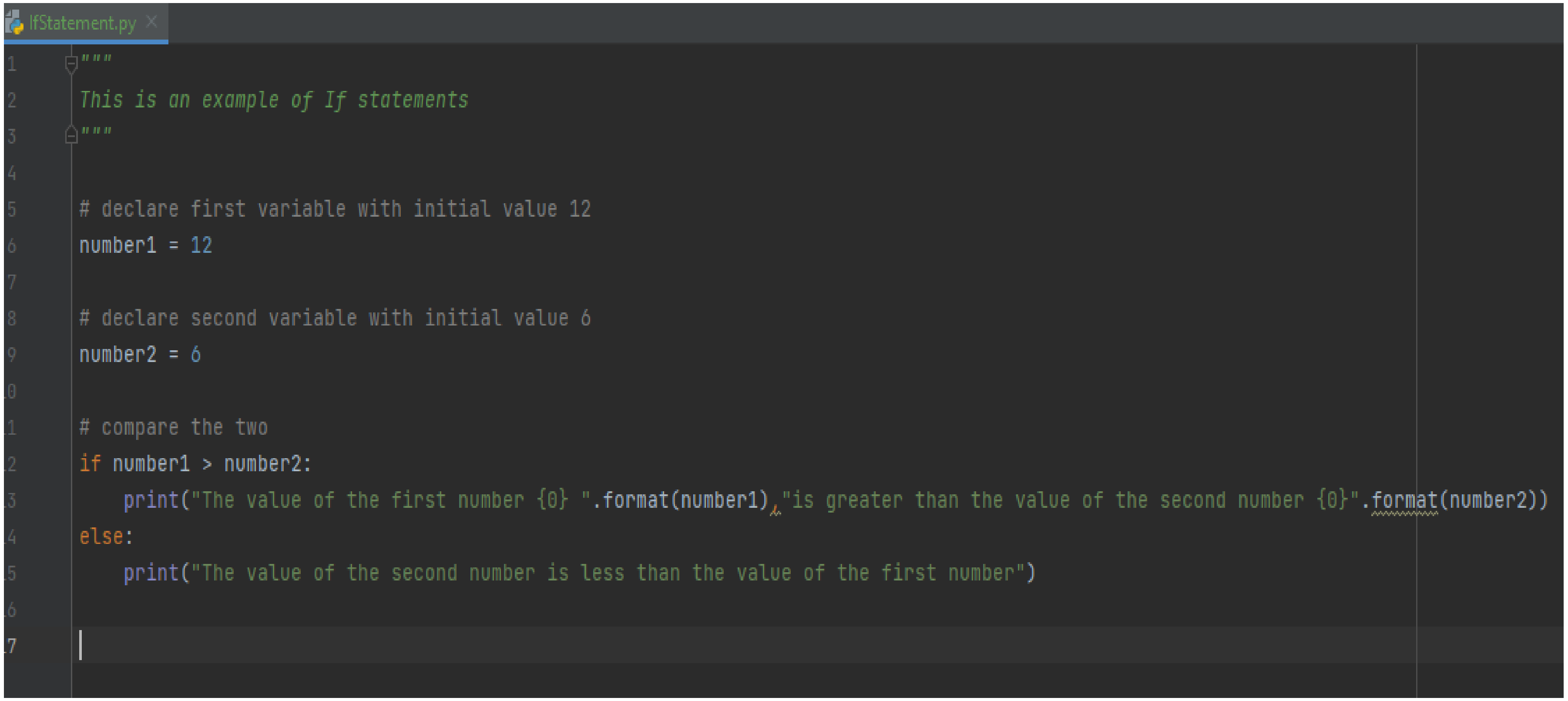
Conditional statements

- One of the most useful operations in programming is if statements.
- Simply, you are testing whether a condition is true or false and act accordingly.
- In your own repositor, create a new python file named IfStatementExample and provide the code in the next slide:

Conditional statements: Example

- One of the most useful operations in programming is if statements.
- Simply, you are testing whether a condition is true or false and act accordingly.
- See example next slide. Also provided on GitHub

Conditional statements: Example


IfStatement.py

```python
"""
This is an example of If statements
"""

# declare first variable with initial value 12
number1 = 12

# declare second variable with initial value 6
number2 = 6

# compare the two
if number1 > number2:
    print("The value of the first number {0} ".format(number1),"is greater than the value of the second number {0}".format(number2))
else:
    print("The value of the second number is less than the value of the first number")

```

West Virginia University
JOHN CHAMBERS COLLEGE OF
BUSINESS AND ECONOMICS

Debugging code

- It is preferable to Debug your code before running. This will allow you to do two main things:
  - Keep track of variables and their values
  - Spot errors as they occur
- Let us debug HelloWorld3.py:
  - Notice the variables names as they change
- We can add breakpoints as well.

Simple Calculator

- Let us create a simple calculator which supports the four main operations (+, -, *, /) between two numbers.
- We want the user to enter the numbers and the operation.
- Can you think of a pseudo code for that?

Simple Calculator

- Do you want the user to provide the program with the entire formula at once? (e.g., 1 + 3)?
- Do you want to ask the user for each (1st value, 2nd value, and the operation)?
- How would you decide what you should do after that? Assuming you have the numbers and the operation, what is next?

Simple Calculator: pseudo code

```
Begin
        Display welcome message
        Ask user for first value -> store in val1
        Ask user for the operation -> store in op
        Ask user for the second value -> store in val2
        Check the value of the op:
        if op is +, then print val1 + val2
        else if op is -, then print val1 – val2
        else if op is /, then print val1 / val2
        else, then print val1 * val 2
End
```

String manipulation

- So, let us think about this: If the equation is given to you, how would you generate the output? What do you need to ask? Can we “extract” difference pieces of information from the given formula?
- How can we extract the first number?
- How can we extract the second number?
- How can we extract the operator?

String manipulation

- Search the web for extracting characters/ words from text in python.
- Use those commands to create a new calculator.
- Name your new calculator FancyCalculator.py, commit and push your code to your repo.

Repetitions

- In all programming languages, there are techniques used to repeatedly do something.
- Think about this, how would you print your name 10 times?
- How do we design a program that keeps asking user for an input “e.g., password” until they provide the correct password?

For loops

- For loops are designed to keep a certain code/ algorithm running until something happens.
- Logical Syntax:

```
As long as a condition is true:
    do something
```

- For loop syntax:

```
for (condition:
        do something.
```

For loops: Example

Please refer to the main repository for example.

Code Documentation

- When you write code, you need to provide meaningful explanation of what you are doing.
- It is highly encouraged (If not required) to provide comments using # and then provide your explanation.
- # adds one line of comment, where as providing “”” (three “) and then enter provides a block of comments. See the codes on 493_Repo for examples

Functions/ Methods

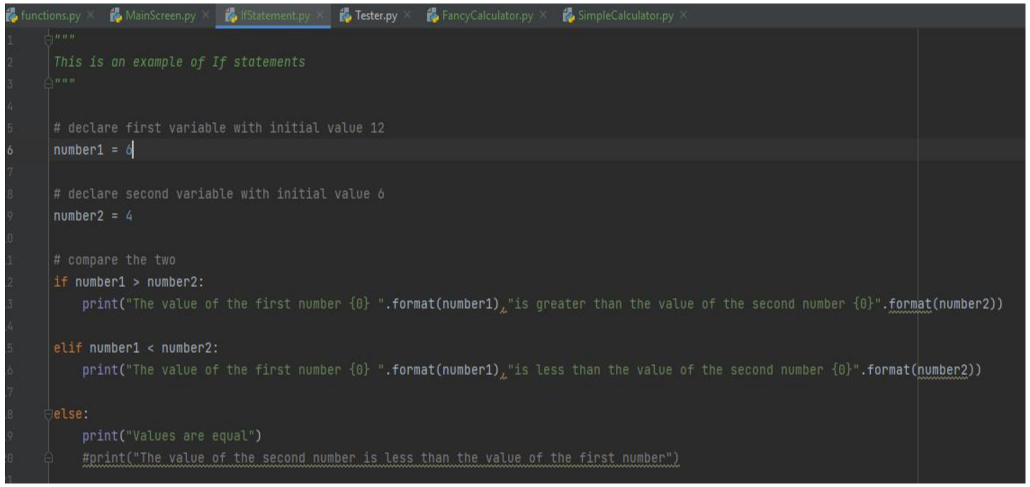
- We have been writing programs in python files, then we run each file to execute the code.
- How can we use these programs from other places?
- What do we need to do if we need the same program to be executed "on demand" as needed?

Methods

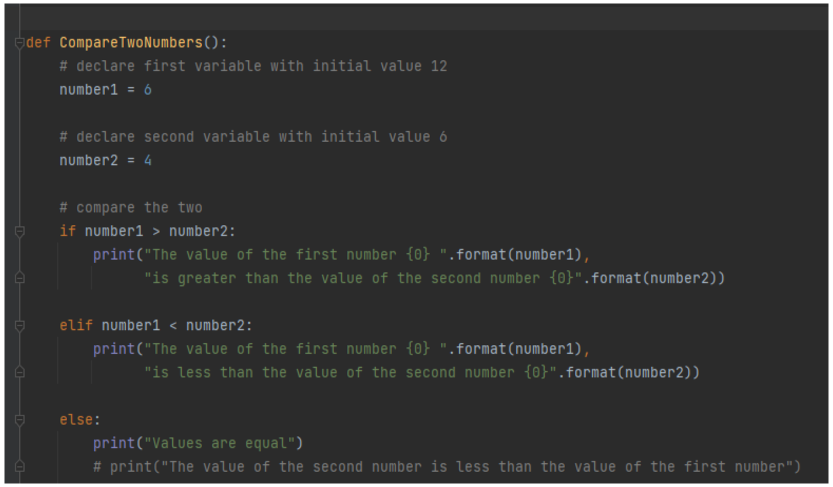
- Methods (also known as function) is simply a way of grouping certain programs under one package with a name.
- Think of the two calculators we made, we can only run one of them at a time, but we can modify them and then call them from different python codes.

Create methods in python

- Simply, start each program with “def [program name()”: Compare the screenshot below:



```python
"""
This is an example of If statements
"""

# declare first variable with initial value 12
number1 = 6

# declare second variable with initial value 6
number2 = 4

# compare the two
if number1 > number2:
    print("The value of the first number {0} ".format(number1),"is greater than the value of the second number {0}".format(number2))

elif number1 < number2:
    print("The value of the first number {0} ".format(number1),"is less than the value of the second number {0}".format(number2))

else:
    print("Values are equal")
    #print("The value of the second number is less than the value of the first number")
```



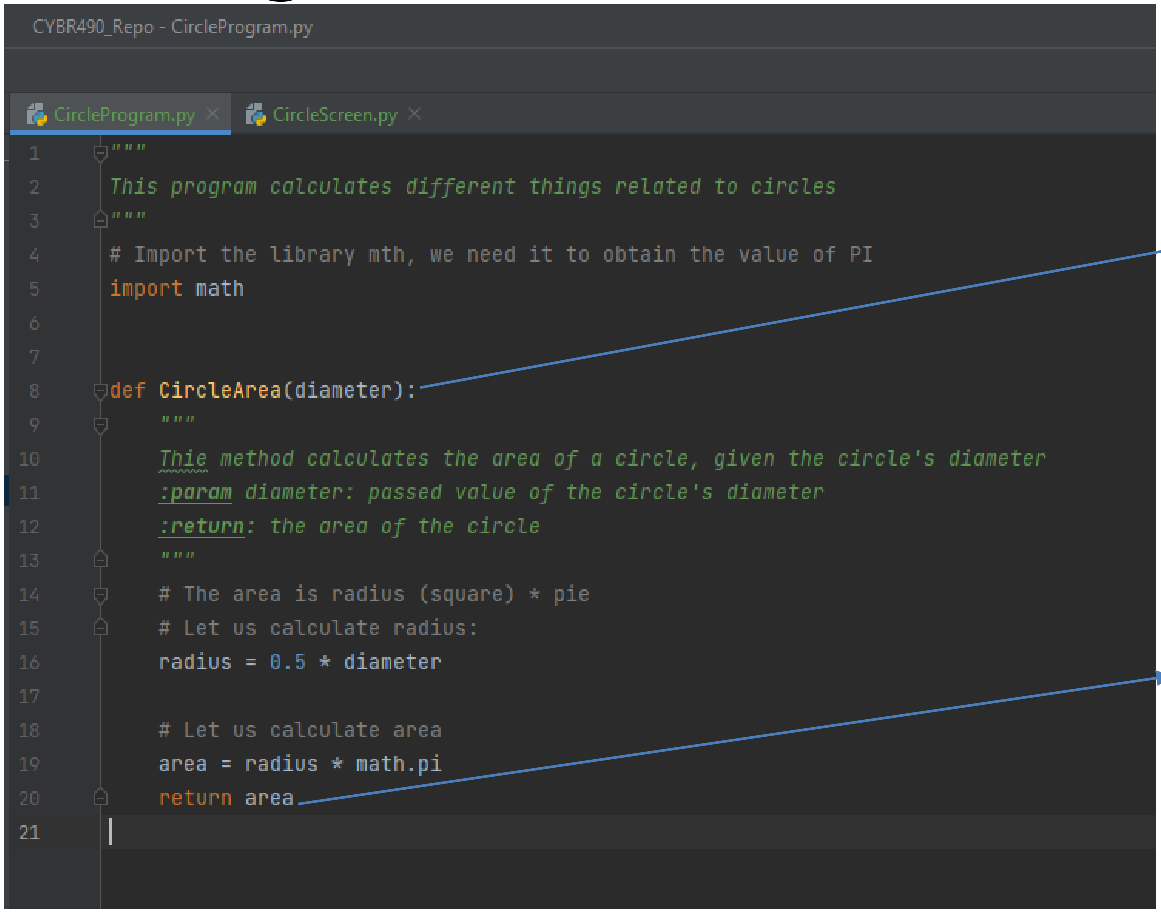
```python
def CompareTwoNumbers():
    # declare first variable with initial value 12
    number1 = 6

    # declare second variable with initial value 6
    number2 = 4

    # compare the two
    if number1 > number2:
        print("The value of the first number {0} ".format(number1),
              "is greater than the value of the second number {0}".format(number2))

    elif number1 < number2:
        print("The value of the first number {0} ".format(number1),
              "is less than the value of the second number {0}".format(number2))

    else:
        print("Values are equal")
        # print("The value of the second number is less than the value of the first number")
```

WestVirginiaUniversity
JOHN CHAMBERS COLLEGE OF
BUSINESS AND ECONOMICS

Passing data to method

- Method are independent but often we need to pass data to them to work.
- Think of a method named CircleArea, which calculates the area of a circle with diameter d.
- The code in the method would look like code in the next slide:

Passing data to method



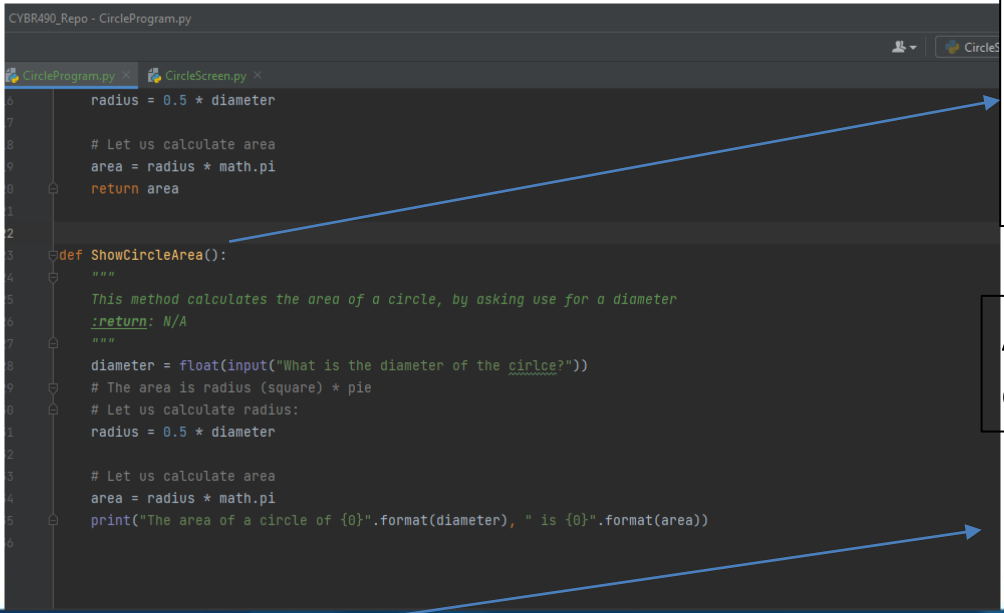
```python
"""
This program calculates different things related to circles
"""
# Import the library mth, we need it to obtain the value of PI
import math


def CircleArea(diameter):
    """
    Thie method calculates the area of a circle, given the circle's diameter
    :param diameter: passed value of the circle's diameter
    :return: the area of the circle
    """
    # The area is radius (square) * pie
    # Let us calculate radius:
    radius = 0.5 * diameter

    # Let us calculate area
    area = radius * math.pi
    return area
```

A method named CircleArea, which accepts one parameter named diameter

After the area is calculated, it gets returned to where its being called from

Methods without passing data

- Some method may not need data passed. Meaning you may accept data inside the method itself.



```
    radius = 0.5 * diameter

    # Let us calculate area
    area = radius * math.pi
    return area


def ShowCircleArea():
    """
    This method calculates the area of a circle, by asking use for a diameter
    :return: N/A
    """
    diameter = float(input("What is the diameter of the cirlce?"))
    # The area is radius (square) * pie
    # Let us calculate radius:
    radius = 0.5 * diameter

    # Let us calculate area
    area = radius * math.pi
    print("The area of a circle of {0}".format(diameter), " is {0}".format(area))
```

A method named ShowCircleArea, which does not accept any parameters

After the area is printed out inside the method

WestVirginiaUniversity
JOHN CHAMBERS COLLEGE OF BUSINESS AND ECONOMICS

Method: types

- There are two types as shown before:
  - Return type: this method does its job and then returns a value (i.e., the result) back to where it is called from.
  - Non-return: This method does not return anything. In most cases, it just printouts out results, or modifies some global variables.