PYTHON SAMPLES

FOR BEGINNERS

WORKING WITH NUMBERS AND OPERATORS

- Different types of operators
  - Multiplication: *
  - Division: /
  - Addition: +
  - Subtraction: -
- Different types of operators
  - Exponents: **
  - Modulus: %
  - Floor: //
- Order of Operations
  - Parenthesis – Exponents – Multiplication – Division – Addition – Subtraction

WORKING WITH NUMBERS AND OPERATORS (PRACTICE)

- print(5+2) = ?
- print(5*2) = ?
- print(5**2) = ?
- print(5/2) = ?
- print(5+2) = ?
- print(5–2) = ?

- print(5//2) = ?
  - Floor: does not provide the remainder
- print(5%2) = ?
  - Modulus: provides the remainder
- Working with large numbers:
  - print(1,000,000)
  - print(1000000) instead

WORKING WITH TEXTS AND STRINGS

- You may use single or double quotes
- Example:
  - “” or ‘’
- Practice
  - print(‘whats up?’): Use of single quotes
  - print(“What’s Up?”): Use of double quotes

WORKING WITH COMMENTS

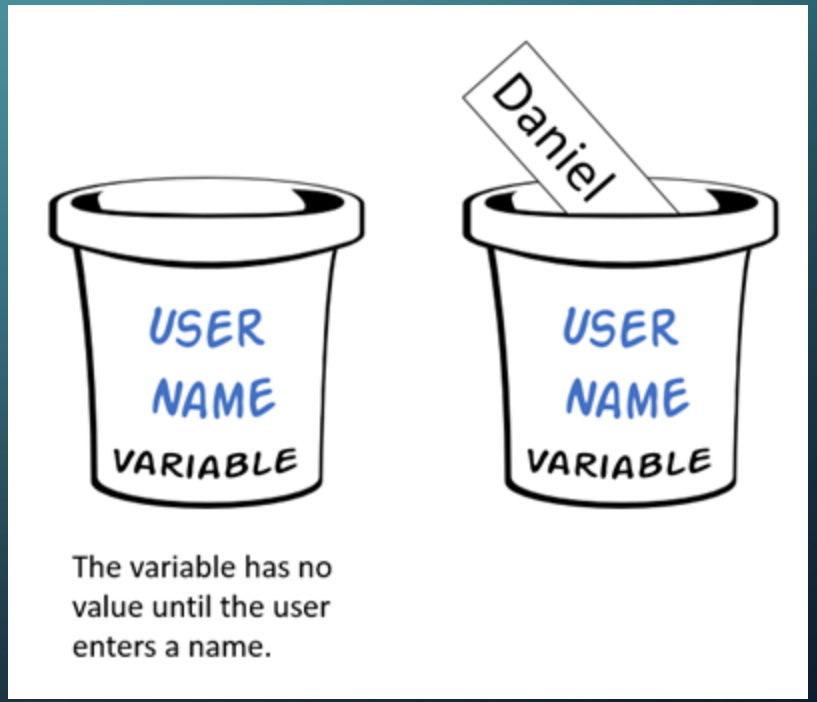
- Use # in order to comment out lines of code
- Python does not execute commented lines
- Practice
  - print("What's Up? ") #comment to describe code
  - Comment out lines of code
    - # print("What's up? ")

WORKING WITH VARIABLES

- Variable: a variable is a value that can change, depending on conditions or on information passed to the program
  - Must contain letters, numbers, or underscores
  - Should not start with a number
  - Spaces are not allowed
  - Cannot be keywords (e.g, print, try, break)
  - Short & descriptive are best (e.g., first_Name)
  - Case sensitive

WORKING WITH VARIABLES

- Algebra: X = 24
  - X = “String”
  - X = Integer



WORKING WITH VARIABLES (PRACTICE)

- bucket = “Eric”
- print(Bucket)

WORKING WITH VARIABLES (PRACTICE)

- bucket = “Eric”
- bucket = 10
- print(Bucket)

WORKING WITH VARIABLES (PRACTICE)

- Data Types: the classification or categorization of data items
  - Integer → int (whole numbers)
  - Strings → str (texts)
  - Boolean → 1 or 0, True of False
  - Floats → Decimal

WORKING WITH VARIABLES (PRACTICE)

- bucket = “Eric”
- bucket = 10
- print(type(bucket))

WORKING WITH VARIABLES (PRACTICE)

- Input(): Takes input from the user and stores it into a variable
- bucket = input("What is your name? ")
- print(bucket)

WORKING WITH VARIABLES (PRACTICE)

- #Input(): Takes input from the user and stores it into a variable
- bucket1 = input("Enter first number? ")
- bucket2 = input("Enter second number? ")
- print(bucket1 + bucket2)

WORKING WITH VARIABLES (PRACTICE)

- #Input(): Takes input from the user and stores it into a variable
- bucket1 = input("Enter first number? ")
- bucket2 = input("Enter second number? ")
- print(int(bucket1) + int(bucket2))

WORKING WITH CONDITIONAL LOGIC

- Conditional Operators: Are the conditional expressions used to give the output according to the condition in the expression

Symbol	Operation
==	Equal to
!=	Not equal to
<	Less than
>	Greater than
>=	Greater than or equal to
<=	Less than or equal to

WORKING WITH CONDITIONAL LOGIC (PRACTICE)

- Conditional Operators
  - print(5 == 4) →
  - print(5 != 4) →
  - print(5 < 4) →
- Conditional Operators
  - print(5 > 4) →
  - print(5 >= 4) →
  - print(5 <= 4) →

WORKING WITH CONDITIONAL LOGIC (PRACTICE)

- Assign some variables:
  - User_Age = 24
  - Age_President = 35
- Compare Variables:
  - print(User_Age == Age_President)
  - print(User_Age != Age_President)
  - print(User_Age > Age_President)
  - print(User_Age < Age_President)
  - print(User_Age >= Age_President)
  - print(User_Age <= Age_President)

WORKING WITH IF STATEMENTS

- Decides whether certain statements need to be executed or not.

WORKING WITH IF STATEMENTS (PRACTICE)

- Assign some variables:
  - User_Age = 24
  - Age_President = 35

  A formatted string literal or f-string is a string literal that is prefixed with f or F . These strings may contain replacement fields, which are expressions delimited by curly braces {}

- Compare Variables:
  - if User_Age < Age_President:
  - print(f"You are {User_Age}, and you're too young to run for president")
  - print("You are", user_age, "and you're too young to run for president")

WORKING WITH IF STATEMENTS (PRACTICE)

- Assign some variables:
  - User_Age = 24
  - Age_President = 35

  An else statement **contains the block of code that executes if the conditional expression in the if statement resolves to 0 or a FALSE value.** The else statement is an optional statement and there could be at most only one else statement following if

- Compare Variables:
  - if User_Age < Age_President:
    - print(f"You are {User_Age}, and you're too young to run for president")
  - else:
    - print(f"You are {User_age} and old enough to run for president or another office")

WORKING WITH IF STATEMENTS (PRACTICE)

- Assign some variables:
  - User_Age = 24
  - Age_President = 35

elif short for "else if" and is used when the first if statement isn't true, but you want to check for another condition

- Compare Variables:
  - if User_Age < Age_President:
    - print(f"You are {User_Age}, and you're too young to run for president")
  - elif User_Age == Age_President:
    - Print(f“Enjoy running for the presidencyat {User_age}”)
  - else:
    - print(f"You are {User_age} and old enough to run for president or another office")

WORKING WITH FUNCTIONS

- A function is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a function.
- A function can return data as a result.
- Can be called throughout the body of code
- Examples of built-in functions:
  - print()
  - input()

WORKING WITH FUNCTIONS

- print(“roadrunners are the best”)
- print(“roadrunners are the best”)
- print(“roadrunners are the best”)

WORKING WITH FUNCTIONS

- Step 1: Define a function
  - def my_function():

- Step 2: Call a function
  - my_function()

WORKING WITH FUNCTIONS (PRACTICE)

- Before
  - print("roadrunners are the best")
  - print("roadrunners are the best")
  - print("roadrunners are the best")

- After: Define and Call
- Define the function first
  - Def my_function():
    - text = "Roadrunners are the best"
    - Print(text)
    - Print(text)
    - Print(text)
- Call the function second
  - my_function()

WORKING WITH FUNCTIONS (PRACTICE)

- Before
- Def my_function():
  - text = (“roadrunners are the best”)
  - Print(text)
  - Print(text)
  - Print(text)
- my_function()

- After: Passing values for arguments
- Def my_function(text):
  - Print(text)
  - Print(text)
  - Print(text)
- my_function(“Roadrunners beat the Redhawks!”)

WORKING WITH FUNCTIONS (PRACTICE)

- Before
- Def my_function(text):
  - Print(text)
  - Print(text)
  - Print(text)
- my_function("Roadrunners beat the Redhawks!")

- After: Returning Values
- Age = input("How old are you? ")
- driving_Age = 16

- def my_function(Age):
- age_diff = driving_Age - int(Age)
- return age_diff

- Num_of_years = my_function(Age)
- print(f"Number of {Num_of_years} to wait until you can drive")

WORKING WITH LOOPS

- For Loop: A for loop in Python is used to iterate over a sequence (list, tuple, set, dictionary, and string).
- While Loop: The while loop is used to execute a set of statements as long as a condition is true.
- Nested Loop: If a loop exists inside the body of another loop, it is called a nested loop

WORKING WITH LOOPS

- While Loop:
  - X = 0
  - while (X<5):
    - print(X)
    - X = X + 1

WORKING WITH LOOPS

- While Loop:
  - X = 0
  - while (X<5):
    - print(X)
    - X = X + 1

- For Loop:
  - Range(start, stop, increment)
  - for X in range(10, 25)
    - print(X)
  - for X in range(10, 25)
    - print(X + 1) → to include last one
  - for X in range(10, 25, 5)
    - print(x)

WORKING WITH LOOPS

- For Loop Integers:
  - Range(start, stop, increment)
  - for X in range(10, 25)
    - print(X)
  - for X in range(10, 25, 5)
    - print(x)

- For Loop Strings:
  - fruits = ["apple", "banana", “grapes"]
    for x in fruits:
      print(x)

WORKING WITH IMPORT STATEMENTS

- In Python, you use the “import” keyword to make code in one module available in another
- Using imports properly will make you more productive, allowing you to reuse code while keeping your projects maintainable
- You need to use the import keyword along with the desired module name. When interpreter comes across an import statement, it imports the module to your current program. You can use the functions inside a module by using a dot(.) operator along with the module name.
- The import statement allows you to import all the functions from a module into your code. Often, though, you'll only want to import a few functions, or just one. If this is the case, you can use the from statement

WORKING WITH IMPORT STATEMENTS (PRACTICE)

- Example:
    - import math
    - Print("Pi is", math.pi)

WORKING WITH IMPORT STATEMENTS (PRACTICE)

- Example:
  - import math
  - from datetime import date
    - Print("Pi is", math.pi)
    - Print("Today's date is", date.today()) #get system date

WORKING WITH IMPORT STATEMENTS (PRACTICE)

- Example:
  - import socket
  - hostname = socket.gethostname()
  - MyIP = socket.gethostbyname(hostname)
  - print(f"My IP address is {MyIP} and my computer name is {hostname}")

FIRST PROGRAM ATTEMPT

- Next Steps:
  - Look to Moodle for your first program assignment