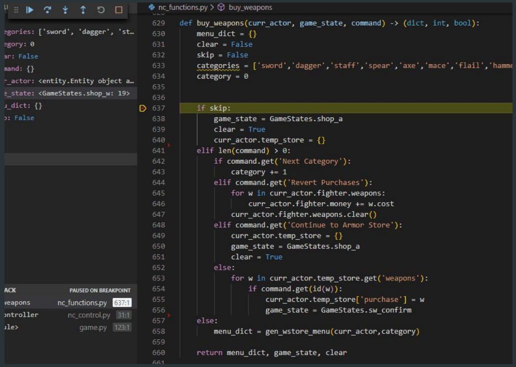
Workshop Introduction To Python

Sign In:

Why Python?

- Python works on different platforms (Windows, MacOS, Linux)
- Syntax similar to the English language (Readability)
- Extensive libraries/frameworks
- Strong industry adoption
- Versatility



```
egories: ['sword', 'dagger', 'st…
egory: 0
ar: False
mand: {}
r_actor: <entity.Entity object a…
e_state: <GameStates.shop_w: 19>
u_dict: {}
p: False
```

```
ACK                    PAUSED ON BREAKPOINT
weapons          nc_functions.py  637:1
ontroller        nc_control.py    31:1
ule>             game.py          123:1
```

```python
def buy_weapons(curr_actor, game_state, command) -> (dict, int, bool):
    menu_dict = {}
    clear = False
    skip = False
    categories = ['sword','dagger','staff','spear','axe','mace','flail','hamme
    category = 0


    if skip:
        game_state = GameStates.shop_a
        clear = True
        curr_actor.temp_store = {}
    elif len(command) > 0:
        if command.get('Next Category'):
            category += 1
        elif command.get('Revert Purchases'):
            for w in curr_actor.fighter.weapons:
                curr_actor.fighter.money += w.cost
            curr_actor.fighter.weapons.clear()
        elif command.get('Continue to Armor Store'):
            curr_actor.temp_store = {}
            game_state = GameStates.shop_a
            clear = True
        else:
            for w in curr_actor.temp_store.get('weapons'):
                if command.get(id(w)):
                    curr_actor.temp_store['purchase'] = w
                    game_state = GameStates.sw_confirm
    else:
        menu_dict = gen_wstore_menu(curr_actor,category)

    return menu_dict, game_state, clear
```

IDE

- Integrated Development Environment or an IDE is a code editor created to easily write, run, and debug code
- Ex: VsCode & Pycharm
- Download or run your preferred IDE to follow along with the workshop

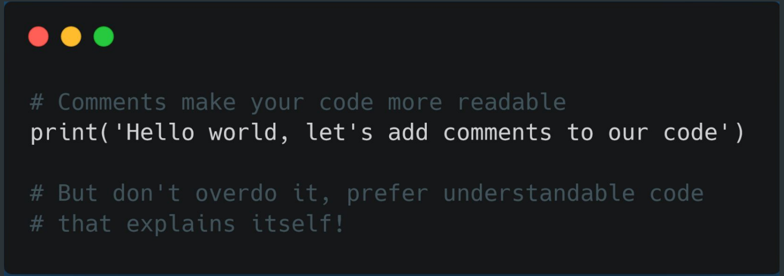
Variables

Camelcase Ex: variableName1

- Containers for storing data values
- Unlike other languages, python does not have a command for assigning variables
- One is created the moment you assign a value to it
- Variable names have to be written in camelcase

Comments

- Comments are text that is not considered to be code
- Used for documentation
- One line comment with #
    - Ex: #one line comment
- Multi line comment or multiline string with “”””
    - Ex: “””” multi line comment “”””

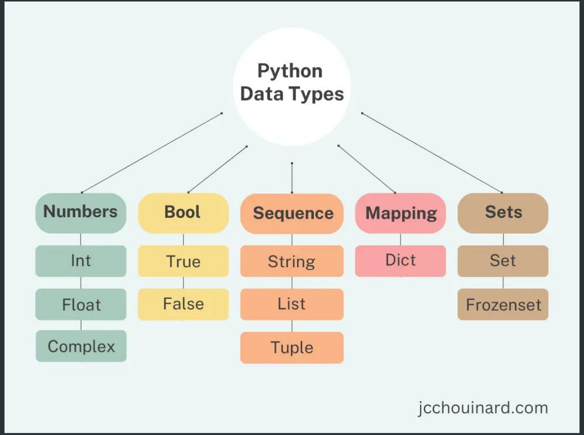


```
# Comments make your code more readable
print('Hello world, let's add comments to our code')

# But don't overdo it, prefer understandable code
# that explains itself!
```

Data Types

- Text = str
- Numerals = int, float, complex
- Sequences = list, tuple, range
- Mapping = dict
- Sets = set, frozenset
- Booleans = bool
- Binary = bytes, bytearray, memoryview
- None = NoneType



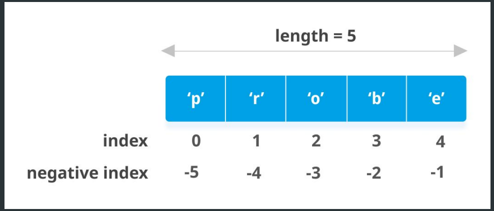
Out/In

- To output information to the system, use the print() method
  - Ex: print(“Hello World”) or print(variableName1)
- To input information to the system use the input() method
  - Ex; password = input(“Enter password:”)
- You can format print statements
  - Ex: print(f“Total points: {pointVar}”) or print(f”Total money: ${money:.2f}”)

Complete Q1.py in your IDE

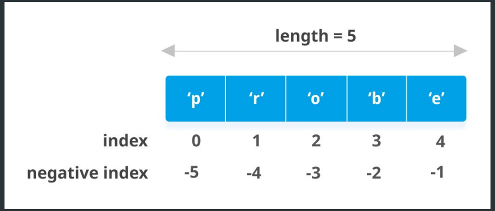
Lists

- Index of list to retrieve & change elements
  - list[0] is the first element
  - list[0] = “new element”
- Slice or Step through list to retrieve elements from an index
  - list[:5] is the first five elements
  - list[::]
- List methods
  - list.append(“New element”) #add new item to list
  - list.remove(“element”) #remove item from list
  - list3 = list1 + list 2 #combine two lists together



Dictionary

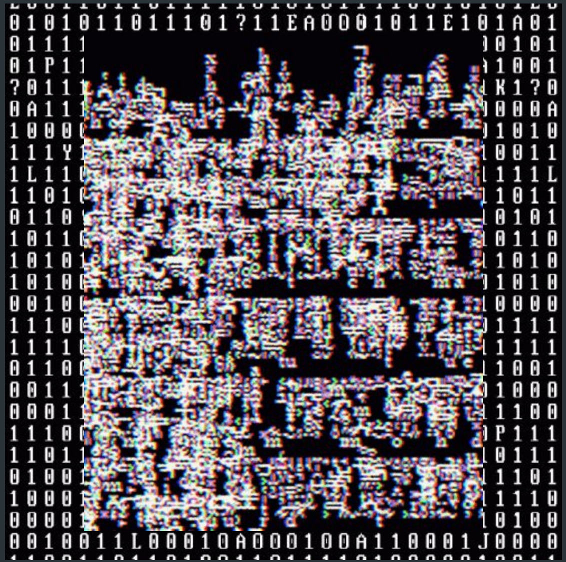
- No duplicate keys allowed in dictionaries
- Dictionary methods
  - dict = {}
  - dict = {key1: value1, key2: value2, …}
  - dict[key] = value
- Key of dictionary to retrieve value pair
  - dict[key] or dict[key1, key2] will give the value/values of a key



Complete Q2.py in your IDE

Operators pt. 1

+	addition	x + y
-	subtraction	x - y
*	multiplication	x * y
/	division	x / y
%	modulus	x % y
**	exponentation	x ** y
//	floor division	x // y



Operators pt. 2

min()	minimum	min(x)
max()	maximum	max(x)
abs()	absolute value	abs(x)
pow()	x^y	pow(3,4) = 3^4

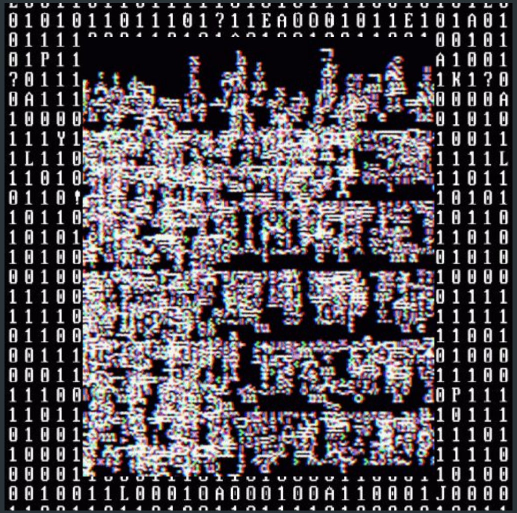
Import math (module)

math.sqrt	square root	math.sqrt(x)
math.pi	PI value	math.pi (3.14…)

Numerical variables in python can be incremented or decremented by a set value using any of the operators

```
x += 2 #increment

y -= 1 #decrement
```



Complete Q3.py in your IDE

Conditionals

If…Else Statements:

- Use logical conditions for scenarios such as if statements where the program will only run the code within the if statement when the set condition is true
- Else statements are run in the scenario where the condition in the if statement is false

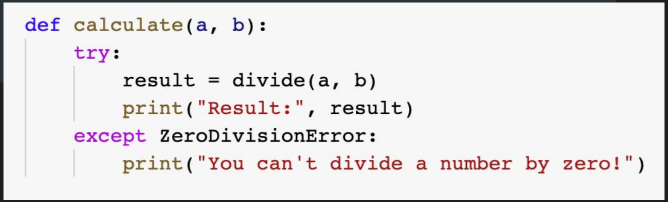
Conditions:

- Equals a == b
- Not Equals a != b
- Less than a < b
- Less than or equal to a <= b
- Greater than a > b
- Greater than or equal to a >= b

Complete Q4.py in your IDE

Try…Except

- The try…except statement follows a similar pattern as an if…else
- The try statement lets the user test a section of code for errors
- The except statement lets the user handle the error



```
def calculate(a, b):
    try:
        result = divide(a, b)
        print("Result:", result)
    except ZeroDivisionError:
        print("You can't divide a number by zero!")
```

Complete Q5.py in your IDE

Loops

- Python includes two types of loop statements
  - While loops execute the code within as long as the set condition is true
    - while x <= y:
  - For loops iterate through a sequence such as a list (works as an iterator method)
    - for element in list:
- Loop commands
  - continue = stop the current iteration and start the next iteration
  - break = stop the loop even in the event where the condition is true
  - range() = iterate through a sequence for a specified range in a for loop
  - pass = skip iteration in loop to avoid error
- Nested Loops are loops within loops
  - Inner loops will run through iterations for each individual iteration of the outer loop

Complete Q6.py in your IDE

Functions

- Block of code that is run only when it is called is a Function
  - Ex: def my_function(): #to create a function
  - Ex: x = my_function() #to call a function
- Pass data arguments as parameters of function to use in code
  - Ex: my_function(x, y): #to pass parameters in function within parentheses
  - Ex: my_function(x = 2, y = “name”): #to pass default values to parameters
- Must be called with the correct amount of arguments
- Can pass any data types as arguments of a function
- return command to output data back to system

Scope

- Global Scope
  - Variables/Data Types that are created outside of functions are able to be used within the scope of any function
- Local Scope
  - Variables/Data Types that are created within functions are only usable within the function it was created in, unless it is returned/output by the function

Complete Q7.py in your IDE

Classes

- Python is an Object oriented program
- Everything used beforehand is an object that contains properties and methods
- Classes act as blueprints in creating new objects, an object constructor
    - class MyClass: #to create a class
    - newObj = MyClass() #to create an object of a class
    - newObj.classMethod() #to use a method within a class
- Initialize class with __init__() function to assign values to object properties

Complete Q8.py in your IDE

Files

- Able to open files within program with the ability to read, write, and create files
  - Ex: f = open(“Filename.txt”)
  - "r" - Read - Default value. Opens a file for reading, error if the file does not exist
  - "a" - Append - Opens a file for appending, creates the file if it does not exist
  - "w" - Write - Opens a file for writing, creates the file if it does not exist
  - "x" - Create - Creates the specified file, returns an error if the file exists
- Use close() command to exit file after program execution is done
  - Ex: f.close()

Complete Q9.py in your IDE

Modules & Libraries pt.1

- Modules are files containing functions & variables that you want to include in your program (may also be considered as libraries)
    - import myModule
    - myModule.name(“Jane Doe”)
    - myModule.list2[0]
- Important Modules/Libraries
    - datetime
    - math
    - json
    - pytorch
    - pandas
    - numpy
    - matplotlib

Modules & Libraries pt.2

- How to install libraries that are not built-in?
  - PIP is a package manager for Python packages/modules
  - package includes all files needed for a module
  - open command line interface and use “pip” command to install packages
    - pip install pytorch
  - once installed, use import within program to use module
    - Import pytorch
  - able to use specific parts of module
    - from sklearn.tree import decisiontreeclassifier

Complete Q10.py in your IDE

Congratulations

You

Have Learned

Python Basics

Sign In: