1. Programming Language is way for us to give instruction to computers.
2. Computer speaks 0 1.
3. Programming Languages are human readable they convert in to machine languages.
4. Need a translator to translate programming languages to machine language.
5. Translator are interpreters and compilers.
6. Interpreters read line by line and execute code.
7. Compiler read entire files and translate to machine language.
8. Python use interpreter.
9. We need to download python interpreter.
10. Python language written in C language.
11. Check GitHub python named as cpython.
12. Py → → → Interpreter → → → cpython VM→ → → Machine Language
13. Python 2 vs Python 3 see the bookmarks
14. Python created by Guido van Rossum in 1991.
15. Python updated in 2008 from v2 to v3. It was a big change because so many enterprise companies were on v2 and Python 2 was crashed.
16. Maintenance was stopped for v2 in 2020 and companies should transfer on v3.
17. Company haven’t upgraded to v3 it is called legacy code.
18. Why python? It is like english language and easy to understand.
19. It is not used developing android applications.
20. It is used for data science, data visualization and web-scrapping.
21. You need to master four things to learn programming Terms, Data Types, Actions and Best Practices.

**Data Types**

**Fundamental Data Types**

1. int
2. float
3. bool
4. str
5. list
6. tuple
7. set
8. dict
9. complex

**Custom Data Types**

1. classes

**Specialized Data Types**

1. We take it from modules.

**None Data Type.**

1. It means nothing.

|  |  |  |
| --- | --- | --- |
| Type | Storage size | Value range |
| char | 1 byte | -128 to 127 or 0 to 255 |
| unsigned char | 1 byte | 0 to 255 |
| signed char | 1 byte | -128 to 127 |
| int | 2 or 4 bytes | -32,768 to 32,767 or -2,147,483,648 to 2,147,483,647 |
| unsigned int | 2 or 4 bytes | 0 to 65,535 or 0 to 4,294,967,295 |
| short | 2 bytes | -32,768 to 32,767 |
| unsigned short | 2 bytes | 0 to 65,535 |
| long | 8 bytes or (4bytes for 32 bit OS) | -9223372036854775808 to 9223372036854775807 |
| unsigned long | 8 bytes | 0 to 18446744073709551615 |

**Numbers Data Types**

1. int

* Numbers without decimal value

1. float

* Numbers with decimal value.

1. Complex

* It is used for complex math operations.

**Math Functions**

* Functions are actions to perform something.
* round() remove the decimal of number. Rounding the value.
* abs() absolute value. No negative number convert negative to positive number.
* bin() It will return binary value of a number.
* Convert a binary in to base 2, int(‘0b101’,2)

**Developer Fundamentals**

* Check other document.

**Operator Precedence**

1. Python follows basic math rules.
2. () → → → → ((20-3)+2\*\*2) → → → First brackets 20-3, power 2\*\*2, then plus 17+4.
3. \*\*
4. /
5. \*
6. -
7. +

**Variables**

1. Variables are ways to store information that can use in program on computer.
2. In python don’t need to define data types with variables like int age = 20 we can do directly age = 20
3. snake\_case
4. Letters, numbers and underscore.
5. Case sensitive.
6. Can not over ride key word.
7. user\_iq and user\_IQ are different.

**Assigning Variables**

iq=80

user\_age = iq/4

a = user\_age

print(a) // it will return 20

**Constant Variables**

1. Constant variables never changes
2. Constant variables written in upper case like PI, VARIABLE.
3. It can change programmatic but the other programmers know it is a constant due to initialization in upper case.

**Dunder Variables**

1. \_\_doc\_\_
2. Shouldn’t use double underscores before variables.
3. Good variable assignment is easy to read program.

**Multiple variable initialization and assignment**

a,b,c = 1,2,3 so,

a=1

b=2

c=3

**Statements VS Expression**

1. iq=100
2. user\_age=iq/5

* Complete line of code is Statement like 1 and 2 separately. These are two statements.
* Iq/5 in line 2 is a expression.

**Augmented Assignment Operator**

some\_value = 5

some\_value = some\_value + 2

* In above lines code changing some\_value result adding 5 in it and it become 7.
* This can simply done using augmented operator.

some\_value = 5

some\_value += 2 //for adding → returns 7

some\_value -= 2 // for subtracting → returns 3

some\_value \*= 2 // for multiplying → returns 10

some\_value /= 2 // for dividing → returns 2.5

**String Data Type**

* String is a piece of text.
* Every thing in double and single quotes are strings.
* For multiple line of strings we use ‘’’ three quotes.
* first\_name = ‘Raheel’
* last\_name = "Ahmed"
* long\_string = """

Checking long strings with lines

yes it is working

"""

**String concatenation (Merging two strings together)**

full\_name = first\_name + " " +last\_name

**Type Conversion**

* print(type(int(100))) // converting 100 into int
* print(type(str(100))) // converting 100 into str
* print(bool(0)) // converting 0 into boolean which will be false
* print(float(1)) // converting 1 into float

**Escape Sequence**

weather = 'It\'s sunny today'

weather1 = "It's cold today"

weather3 = "It's \"kind of sunny\" today"

weather4 = "\tIt is raining" \t used for tab

weather5= "\n new line" \n use for new line

**Formatted String - It is use to avoid concatenation**

name = "Raheel"

age = 30

print(f'My name is {name}. I am {age} years old')

In python 2 used

print('My name is {}. I am {} years old'.format(name, age))

print('My name is {}. I am {} years old'.format('Raheel', 30))

print('My name is {1}. I am {0} years old'.format('Raheel', 30))

print('My name is {new\_name}. I am {new\_age} years old'.format(new\_name = 'Raheel', new\_age = 30))

**String Indexes**

selfish = "12345678"

01234567 indexes

print(selfish[0]) # accessing string on the index 0

**String manipulation**

[start : stop: step-over] it is called slicing

* print(selfish[0:8:1]) # start from 0 index end 8th index and stepover by 1
* print(selfish[0:8:2]) # start from 0 index end 8th index and stepover by 2
* print(selfish[1:]) # start from 1st index and everything default
* print(selfish[:5]) # start from 0 index and end on 5th index and stepover by 1 (default)
* print(selfish[::2]) # start and end are default stepover by 2
* print(selfish[-3]) # start from reverse and take the 3rd
* print(selfish[::-1]) # reverse with stepover by 1

**String Immutability**

* Strings are immutable we can not change string with giving index we have to create new variable for changing it.

sample = "01234567" # old sample

sample[0] = 4 # we can not do this because strings are immutable

sample = sample + '8' # now sample will be this

print(sample)

**Built-In Functions**

* len("hellooo")

it returns length of values return value not start from 0

* Methods are built-in functions as well but Methods are own to something like string methods, number methods
* See documentation for more

**String Methods**

* quote = "to be not to be"
* print(quote.upper()) # Convert string in tp upper case
* print(quote.capitalize()) # Capitalize string first letter
* print(quote.lower()) # Convert string in to lower case
* print(quote.find("be")) # find the starting index of be
* print(quote.replace("be","me")) # it is returning replaced quote be to me but original will be same.

**Boolean**

* There are two options in boolean
  + True
  + False
* Boolean represent True and False which in programming is kind of like 0 and 1.
* Boolean are useful in conditionals
* bool(1) = True
* bool(0) = False
* bool(‘True’) = True
* bool(‘False’) = False

**List**

* We denote list with []
* Collection of items of any data type.
* List are form of arrays
* List is a Data Structure
* li = [1,2,3]
* li2 = [‘a’,’b’,’c’]
* li3 = [1,2,’a’,’b’,True]
* For accessing list we use index li[0].

**List slicing**

* List slicing is just like string slicing.
* li[start:stop:step-over]
* Lists are mutable
* We can change items with their index like li[2]=’notebooks’
* List slicing return new list.

amazon\_cart=[

'notebooks',

'sunglasses',

'toys',

'grapes'

]

amazon\_cart[0]= 'laptop'

Assigning amazon\_cart to new cart list

new\_cart = amazon\_cart

new\_cart[0] = 'gum'

print(new\_cart)

print(amazon\_cart)

For Copying list we use slicing function

new\_cart = amazon\_cart[:]

new\_cart[0] = 'gum'

print(new\_cart)

print(amazon\_cart)

**Matrix**

* A way to describe multi-dimensional list

matrix = [

[1,2,3],

[4,5,6],

[7,8,9]

]

print(matrix[0][1])

List Methods

* basket = [1,2,3,4,5,6]
* len(basket) → It is a function
* basket.append(value)
  + It add an object/item in the end. It doesn't return value changes in original.
* basket.insert(index,object)
  + It is not return anything
* basket.extend([100,101])
  + It takes many items as list. It doesn’t return any thing
* basket.pop()
  + It removes the item in the end. It returns the removed item.
* basket.pop(index)
  + It removes the item with index. It returns the removed item.
* basket.remove(value)
  + It removes by the value. It doesn’t return anything.
* basket.clear()
  + It doesn’t return anything and clear all the items in the list.
* basket.index(value,start,stop)
  + it returns the index of the value.
  + Start is where should start searching
  + Stop is where should stop searching
  + Sometime we don’t know if the value exist in the list if not, it will give an error which is not good so we use “in” key word for searching
  + 4 in basket → it will return True or False
* basket.count(value)
  + it returns the count of the given value.
* basket.sort()
  + It doesn’t return anything. It sorted the list.
* sorted(basket)
  + it return sorted list.
  + It sorted the list and the original list will not change.
* basket.copy()
  + it copies the list it return the copied list
* basket.reverse()
  + it reverse the original list.

**Common List Patterns**

* These pattern are used most
  + len(basket)
  + basket[::-1]
  + range(1,100)
  + list(range(1,100))
  + ‘ ’.join([“hi”,”yes”])
    - it returns object and join the string list. It is used to combine string with list.

**List Unpacking**

* List unpacking assign different variable to each item of the list.
* a,b,c = [1,2,3]
  + a=1, b=2, c=3
* a,b,c, \*others = [1,2,3,4,5,6,7,8]
  + a=1,b=2,c=3,others = [4,5,6,7,8]

**None** **Datatype**

* It represent the absence of value.
* We can not leave a variable empty so we can assign it with None means nothing for now.

Dictionary

* It is a Datatype and Data Structure
* It is way of organizing data.
* It is a data type with key→ value pair
* It is unordered Key value pair
* In dictionary value can be any data type

dictionary={

‘a’:’hello’,

‘b’:[1,2,3],

‘c’:True

}

* Values can be access with the key

print(dictionary[‘b’])

**Dictionary Keys**

* Dictionary keys can be
  + String
  + Number
  + Boolean
  + Tuple
* Keys need to be immutable
* Keys should be unique other wise it will override.

Dictionary Methods

* user.get(‘age’)
  + It is used to access the value of given key.
  + If key doesn’t exist in dictionary program will stop. To avoiding this we do:
    - user.get(‘age’,55)
    - if key doesn’t exist 55 will added as a default value of age
* user = dict(name=’John’)
  + it is function for creating dictionary
* user.keys()
  + It returns all the keys of dictionary.
* user.values()
  + it returns all the values of dictionary.
* ‘basket’ in user
  + It return True of False that string is available or not in the dictionary
* ‘hello’ in user.keys()
  + It return True of False that string is available or not in the keys
* ‘hello’ in user.values()
  + It return True of False that string is available or not in the values
* user.items()
  + returns and Iterate all the items in the dictionary as a form of tuple. Keys and values are return in the tuple.
* user.clear()
  + It clears the original dictionary
* user.copy()
  + It copies the dictionary and returns the new one.
* user.pop(key)
  + It removes the given key value from the original
* user.popitem()
  + It removes the random value from dictionary and returns the value as well.
* user.update({‘key’:value})
  + it updated the given key value in original dictionary.
  + If key is not available it will create a new key and value in the original dictionary

**Tuples**

* Tuples are like list but immutable
* Tuples initialize in () small brackets. my\_tuple = (1,2,3,4,5)
* It can access like list with index my\_tuple[0]
* If programmer don’t want to change any list so can use tuples
* they are faster than list
* we can slice tuples as well
* We can use unpacking in tuples as well
* my\_tuple.count(value)
  + return count the values
* my\_tuple.index(value)
  + return the index of the value
* len(my\_tuple)
  + return length of tuple

Sets

* Unordered collection of unique items.
* my\_set={1,2,3,4}
* set.add(100)
  + it will add 100 in the set.
* set.add(2)
  + it will not add 2 because that is already available.
* convert in to set. Set([8,9,10])
* set can not access by index.
* Sets can access by value and just return True or False. Because they are unique so just yes or not.
* my\_set.copy().
  + Copy and return a new set.
* my\_set.clear().
  + Clear the original set.
* my\_set.discard(value)
  + It remove the value from original set.
* my\_set.difference(your\_set)
  + It returns value which are difference in set
* my\_set.difference\_update(your\_set)
  + It modifies original set which are difference in the sets
* my\_set.intersection(your\_set)
  + It returns common values
* my\_set.isdisjoint(your\_set)
  + It returns True or False
  + If not common values in both set return True.
* my\_set.union(your\_set)
  + It combines both sets and returns a new one.
* my\_set.issubset(your\_set)
  + It returns true or false if the subset.
* my\_set.issuperset(your\_set)
  + It returns true or false if superset.