Class 1

Thursday, August 22, 2024 09:55

Programming Language (opposite of "natural" lang)

↳ Language that comps "speak" → interpret

↳ Syntax & grammar : programming language => very rigid
↳ One mistake, it crashes → bugs, errors, warnings

↳ all software is made from PLs

↳ 1000s of PLs : EG: C, C++, C#, Java, Javascript, python, rust, SQL, etc...
Assembly/Machine Language

↳ End goal : binary (1101101....)

↳ 2 kinds of PL :

↳ Interpreted → reads each line one by one and turns each line into binary
EG PLs : Python, Java, JS

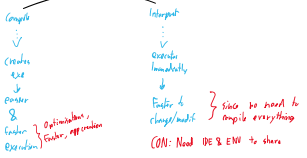
↳ Compiled → uses a Compiler to output a new program in binary
INPUT : Program in given PL
OUTPUT : Executible program in binary (.exe)
EG PLs that use compilers : Java, C, C++, C#

Q : how are compiled and interpreted processes different?
Line by Line VS entire script compiled before exe

Will compiling run a program? Depends... Touch wisteria : No

interpreting? Yes... still depends tho...

Why use one over the other?



✳ A compiler INPUTS and OUTPUTS

What is Python?

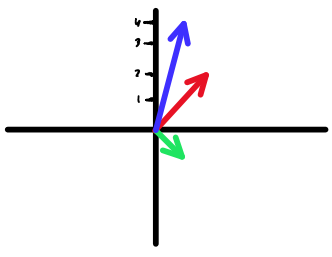
↳ Free & Open-Source Software (FOSS)
↳ Free and always will be
↳ Anyone can look at the code, anyone can contribute ☺
OPPOSITE : closed source, proprietary EG: Microsoft, Spotify
↳ Python = FOSS, but also a company (ppl paid to dev)

↳ Python has many diff versions
↳ Updated over the years
↳ python V1 V2 V3 Big diff between them.
Same Same But Different

↳ tons of applications
↳ Science, data science, AI, Industry, health dist, ...
↳ Big PROs : easier, syntax isn't so bad

Data

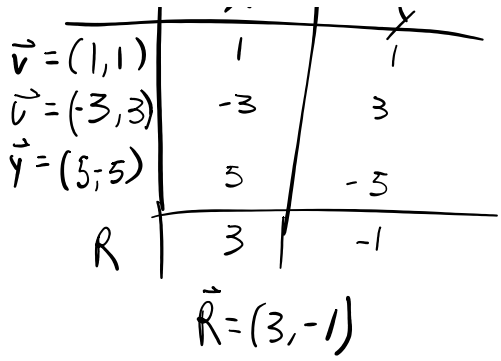
Value	Type
EG: 4	Eg: int
4.76	float
"Hello WORLD"	str



y ** 0.5

y ** 2

	x	y
$\vec{v} = (1,1)$	1	1


x
$\vec{v} = (1,1)$ 1 1
$\vec{v} = (-3,3)$ -3 3
$\vec{y} = (5,-5)$ 5 -5
R 3 -1
$\vec{R} = (3,-1)$

Class 2-3

August 26, 2024 3:33 PM

Look up table

Var name	Value
x	4
y	8
bean_count	28.66
Fav_word	"choose"

Expression = eval one value at a time until value is reached

Expression = line of code that can be simplified

"the process of by-hand stepping through the eval is called Tracing"

Expressions : line of code that can be simplified
Evaluation : ONE step of the simplification
Tracing : Simplifying an expression (Whole process)

} Exam Vocab

Tracing EG:

1. x=4
2. x = x + x * also written x += x
   ⇓ x = 4 + 4 * Substitution : replace vars with their values
   ⇓ x = 8 * Substitution
   PEMDAS
   Assignment : update var table

Class 7 : Pop Quiz at start :

Types :

All data of a computer (text, numbers, pictures)
has 2 elements, 2 sides of a single coin

Data

	Value	Type
EG:	4	int
	2.0	float

As you can imagine, Python, like a calculator, can support whole numbers (int) and decimal numbers (float)

After a float - int calculation, Python will output a float

This said...
We can convert data from one TYPE to another TYPE.
HOWEVER we run the risk of losing data

We can ask python, at ANY POINT in our program to identify the type (**USING type()**) and Value (**USING print()**) of a piece of data

"Class" synonym of "Type"

Values : infinite possibilities
Types : finite possibilities

Built-in ways to convert types in Python

INT -> Float

Cheap way :
```
x=4
x = 4 + 0.0
>>> 4.0 : float
```

Developer way :
```
x = 4
x = float(x)
>>> 4.0 : float
```

} This operation is called "Casting"

Developer way: x = 4.0
x = float(x)
>>> 4.0 : float

This operation is called casting

Float → INT

Developer way: x = 4.0
x = int(x)
x >>> 4

Potential Data loss: x = 4.9
x = int(x)
x >>> 4

Python can operate on strings

Strings (str) are text

They "string" characters together

Review

Type: kinds of Data

Values: "constant literals", pure data

Strings

↳ textual Data (Ex: "beans", "Eric", "SF 1"

Motivation for types

↳ Consider '+'

↳ Types help inform the computer how the data should be dealt with

>>> 1 + 1 ⇒ 2

>>> "1" + "1" ⇒ "11" Concatenation

Booleans

↳ 2 possible values: True or False

↳ type(True) returns <class: bool>

↳ type: bool

↳ Introduces tons of new operations:

- OR, AND, XOR, XAND, NOR, NOT, ...
- "==" equality: compares 2 values to see if equals
- "!=" inequality: compares 2 values to see if not equals

Revised OOO (order of operations)

1. Variable substitution
2. PEMDAS
3. Logical Operators (==, !=, >, <, >=, <=, or, and, etc...)
4. Variable assignment

PRACTICE

1. _____

x = 4
y = False
z = "hi"

a = 4 + 4 > x == y
a = 4 + 4 > 4 == False
a = 8 > 4 == False
a = True == False
a = False

2. _____

s = "hi"

s != s + s != True

"hi" != "hi" + "hi" != True

"hi" != "hihi" != True

True != True

False

True != True != True

"hi" != "hi" + "hi" != True

"hi" != "hihi" != True

True != "hi" != "hihi"
True != True != "hihi"
False != "hihi"
False != True
True

~	

Str -> Bool

"" => False

"{Anything}" => True

Int -> Bool

0 => False

$\mathbb{R}^*$ => True

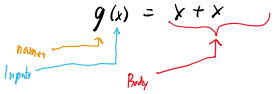
Class 4

August 30, 2024 9:06 AM

Functions:

$f(x) = 2*x + 4$

$g(x) = x + x$



Call the function:

f(5)

To "call" the function, use name and provide **ARGUMENTS**

Above, we call function "f" with args 5

f(5) really means:

- Var assignment -> x = 5
- body -> 2*x + 4

More complex example:

```
>>> def h(x, y): h + y
>>> h(1, 2) => x + y  where x=1 & y=2
            => 1 + 2
            => 3
>>> h(2, 1) => 2 + 1  where x=2 & y=1
            => 3
```

Python Function syntax

```
>>> def addition(x,y):
>>> .... return x + y
>>> ....
>>> addition(3,7)
>>> 7
```

Func won't return anything without syntax word: "return"

Arg-Less func()

```
>>> fun():
      return 2
>>> fun() + fun() -> 4
```

ARGUMENTS:

Type definition of a function:

```
>>> def add(x : int, y : int) -> int:
        return x + y
```

Type annotations, REQUIRED in this course for all inputs & outputs

>>> def [illegible]

return x + y

for all inputs & outputs

>>> add(2,2)

>>> 4 type 'int'

↳ Recall: type definition of a var

EG: add (x:int, y:int) -> int

↳ function -> output Type

EG: add(x:int, y:int) -> int

↳ We can look at a function purely in terms of input & ouput types

↳ add : int (x) -> int (y) -> int (return output)

September 4, 2024 2:45 PM

Review:

parameters => x and y and... in function header definitions

argument => values given to parameters when function called

Syntax:

```
def f(x:int) -> int:
    return 3*x+2
```

x = 0

f(5) > 0+x

Don't forget this whilst Tracing exps

(x = 5 ; 3*(5)+2) > 0+(0)

Submit 10 solutions

Class 6 : if statements

September 6, 2024 9:07 AM

Our code can have branches !!!

↳ if/else statements !!!

Example :

```
1  x = 4
2  if x < 10 :
3      print("hello")
4  else :
5      print("BYE!")
6  print("chicken")
```

True → runs

skipped

not in code block → runs

output

>>> hello

>>> chicken

Why use lists :

- lots of data
- Database
- Counts
- for loops using item database

list = [item 1, item 2, item n]

0 1 2

↳ index

↳ ~~types~~

list = ["Anna", 86]

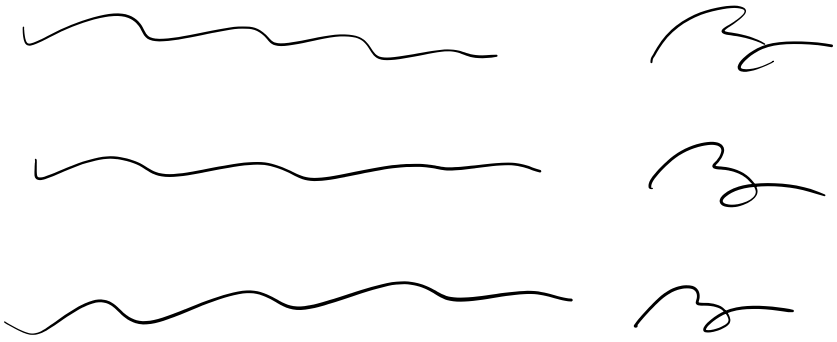
list 2 = ["Bob", 88]

list + list 2

Nanas

>>> ✓

Name: Joseph Score 103/100



Lists - Python

September 13, 2024 08:21

.append() = Add something at the end of the list, or at the index within parenthesis

len(LIST_NAME) = Length of the list, number of values in the list

.remove() = Removes the VALUE in the list, indicated within parenthesis

Slicing: a way to get a subset of a list
Examples:
list[START_OF_SUBSET : END_OF_SUBSET : STEPS] NOTE: END_OF_SUBSET **NOT** INCLUDED. In other words, **"up to, but not included"**

Tools For Software Dev

September 23, 2024 14:35

- Vim: Text Editing
- Command Line (CMD // git bash // linux cmdL): Running and managing files
- Git: Version Control

What is git:

- A technology that helps us:
    1. Distribute files
    2. Keep track of changes
- Version control keeps track of all changes in a file WITH TIMESTAMP logging
- Very helpful for collab
- Git => on the cmdL
- 95% of companies (2024) use Git
- GitHub and GitLab host our modified files on their servers

Git Clone:

- On sites like GitHub and GitLab, people post their code bases
- On the cmdL we can download people's code using `git clone <url to repo>`

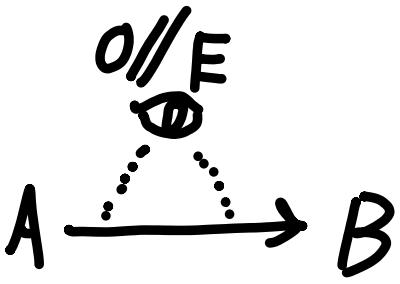
Vim

September 23, 2024 14:36

I hate vim.

Cryptology

September 27, 2024 14:49



Alice (Person A) => Sender
Bob (Person B) => Receiver
Oscar /Eve (Enemy, Opp) => Interceptor

Cryptosystem:
A five tuple (P, C, K, E, D) where the following conditions are satisfied:

- P, the finite set of possible plaintext
- C, the finite set of ciphertext
- K, the keyspace, the finite set of possible keys
- For each x \within K, there is an encryption rune (e of x) \within E and a corresponding decryption rune (d of x) \within D.
    - Every (e of x), P -> C & every (d of x), C -> K
- ASSUMING A SECRET KEY HAS BEEN ESTABLISHED EITHER IN THE PAST OR THROUGH A SEPARATE SECURE CHANNEL

Secret-Key Cryptosystems: key = hidden
Public-Key Cryptosystem: key = clear

Cryptanalysis (THE FUN PART!!!) => Breaking the security key

Public Key Crypting:

- Uses a public key to encrypt, a private key to decrypt

BLOC CYPHER:

- plaintext divided into blocs, encrypted and decrypted accoringly

Stream CYPHER:

- Uses a key to generate a key the same length as the plaintext

EXAMPLE OF CYPHER USING Xor BIT OPERATOR

HYBRID CRYPTOSYSTEM:

A generates random secret key, encrypts her message using that key, encrypts that key using B's public key.
B decrypts A's secret key, decrypts A's message using the deciphered key to get the decrypted plaintext.

Types of adversaries:
Passive ADV:
Reads message

ACTIVE ADV:

- Changes message sent from A to B before B
- Sends fake message to B from A
- Sends message sent from A to C

Secret key setting:
MAC: message authentication code

Public key setting:
Signature scheme purpose

MAC != MAC address (MACA)

SCRTKEY
MAC is like a signature to make sure the message has not been altered during delivery

PUBLKEY
Signing algorithm produces a signature, example: PGP key
CHECK CICADA 1101

ENCRYPTION FUNCTION SHOULD BE INJECTIVE (SEE DISCRETE)

Modulo
Remainder
11 x 13 in Z16
=
143 = 8 x 16 + 15
=
143 mod 16 = 15 Z1

For loops

October 4, 2024 08:57

REVISE THIS SECTION WITH THE POSTED NOTES ON SF1 WEBSITE IN THE NEXT FEW DAYS

Statements:

Side effect:
Printing, something that happens on the side of the code

For loops,

for i in list:

    i = list value correspond for the duration of the function,
    list = any list or equivalent

Range func:

range(START, STOP, STEP)

Outputs:

[start , start + step , start + 2*step , … , start + n*step]
Upto stop and deletes the "stop value"

MIMO for loops & range for extra practice

Assignment 2 structure

October 7, 2024 18:19

win = p1_card && p2_card