



**Programming Principles in Python**

# **Introduction**

---

Samuel A.



belfast  
met

The logo for Belfast Met is a large, stylized blue 'Q' shape. Inside the 'Q', the words 'belfast' and 'met' are written in a blue, lowercase, sans-serif font. The 'Q' is composed of several concentric, slightly curved lines that create a sense of motion or a stylized letter.




# Lecturer

---

Samuel Adebayo

[sadebayo@belfastmet.ac.uk](mailto:sadebayo@belfastmet.ac.uk)



# **WK 1 Lecture 1 Module Outline and Introduction**



# Outline

- **Module Induction**
- Background to module (the why)
- Introduction to Computer Programming
- Basics of programming
- Natural Language
- Translators
- Compilers
- Assemblers



# Arrangements

## Lectures

- Programming Principles in Python      Wk 23 – 38 (Wednesdays)      M-1-4-09

# Arrangements

## Lectures

- |                                    |                         |          |
|------------------------------------|-------------------------|----------|
| • Programming Principles in Python | Wk 23 – 38 (Wednesdays) | M-1-4-09 |
|------------------------------------|-------------------------|----------|

## Labs/Tutorials

- |                              |                      |          |
|------------------------------|----------------------|----------|
| • Codes/Tutorials/Codewithme | Wk 25 – 40 (Fridays) | M-1-4-05 |
|------------------------------|----------------------|----------|

# Arrangements

## Lectures

- Programming Principles in Python      Wk 23 – 38 (Wednesdays)      M-1-4-09

## Labs/Tutorials

- Codes/Tutorials/Codewithme      Wk 25 – 40 (Fridays)      M-1-4-05

**Both Lectures and Labs are compulsory.**

***PCs are available to code on.***

# Assessment- dates to be confirmed

The module is assessed through a continuous assessment which is made up of a coding portfolio, programming assessment, and Assignments.

Week	Assessment	Content	Weightning
4	Assignment 1	Wks 1 -4	10%
6	Assignment 2		15%
8	Assignment 3		20%
9	Class Tests		20%
10	Assignment 4		10%
13	Assignment 5		10%
	Project		15%



# Requirements:

---



Attendance at  
Lectures and Labs.



8 Hours of private  
study per week  
including assessment.



Completion of all  
work as outlined.



Asks questions if you  
don't understand,  
Teams, email etc.



# Outline

- Module Induction
- **Background to module (the why)**
- Introduction to Computer Programming
- Basics of programming
- Natural Language
- Translators
- Compilers
- Assemblers



# Why?

Programming is an essential skill in today's digital age, and studying it can lead to numerous benefits and opportunities:

- **Career Advancement:** Knowledge of programming can lead to lucrative careers in software development, data analysis, and many other technology-related fields.
- **Problem-Solving:** Programming provides a structure for solving complex problems through logic and algorithms.
- **Collaboration:** Programming is often a collaborative effort, allowing you to work with others and learn from their skills and perspectives.



# Why?

Programming is an essential skill in today's digital age, and studying it can lead to numerous benefits and opportunities:

- **Creativity:** Programming allows you to turn your ideas into reality by creating software, websites, and applications.
- **Digital Literacy:** A basic understanding of programming is becoming increasingly important as technology continues to play a central role in society and commerce.

Overall, studying programming can open doors to new and exciting career paths, as well as develop valuable skills in logic, problem-solving, and creativity.



# Outline

- Module Induction
- Background to module (the why)
- **Introduction to Computer Programming**
- Basics of programming
- Natural Language
- Translators
- Compilers
- Assemblers

# Programming

- Programming is the process of designing, writing, testing, debugging, and maintaining the source code of the computer software.
- It involves the use of programming language to create instructions that a computer can understand and execute to perform specific tasks.
- The goal of programming is to develop software that solves problems, automates processes, and enhances human productivity.





# Programming

- Programming also involves problem-solving, critical thinking, and creativity. Programmers work on a wide range of software applications, from simple scripts to complex systems, and from desktop applications to web and mobile apps.
- **Effective Programming!**
- **High Quality Software**
- **Scalable and Reliable**



# Programming: what is required?

- Ability to understand and analyze the problem.
- Design a solution
- Write and test codes
- Debugging
- Maintainability



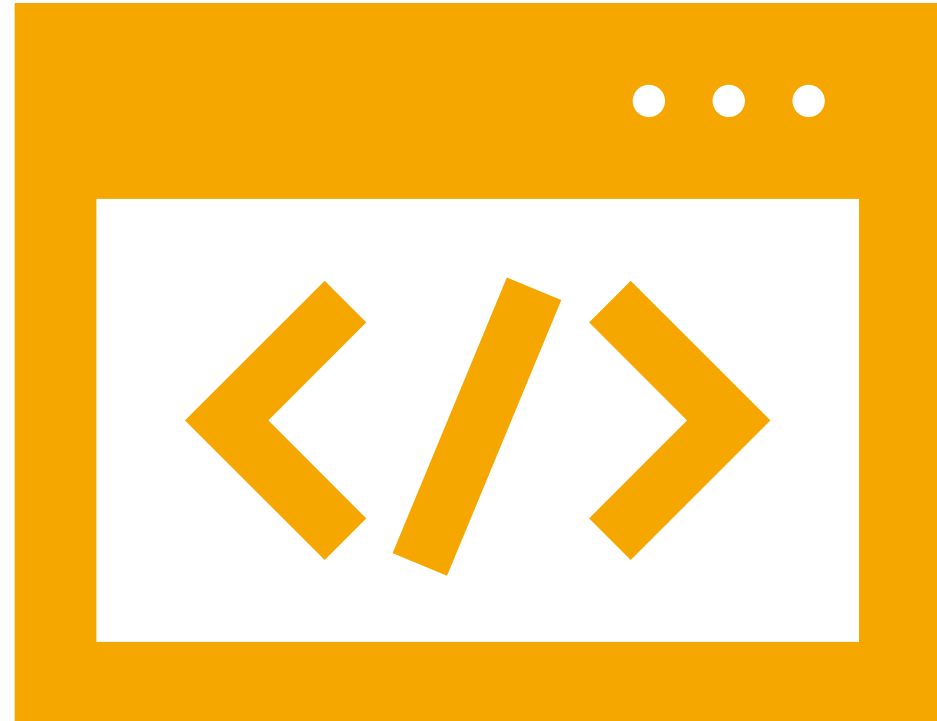
# Types of Programming

- Imperative Programming
- Functional Programming
- Declarative Programming
- Event-driven Programming
- Scripting Programming
- **Object-oriented Programming**

# Imperative Programming

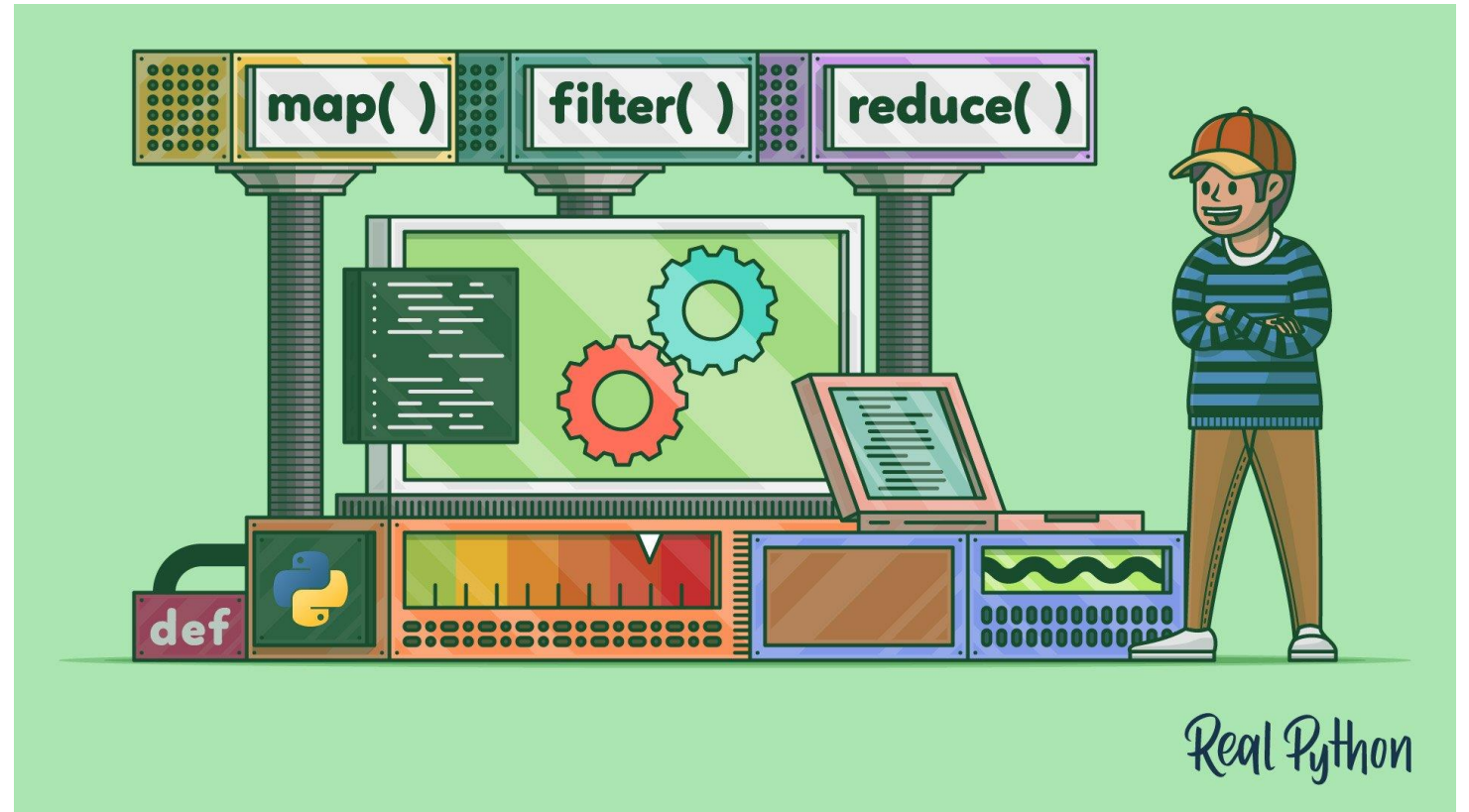
---

This type of programming is based on procedures and sequences of statements that change a program's state. It includes languages such as C, C++, and Pascal.



# Functional Programming

This type of programming is based on mathematical functions and emphasizes the evaluation of expressions rather than the execution of commands. It includes languages such as Haskell, Lisp, and Scheme.



# Declarative Programming

This type of programming focuses on describing the desired results rather than specifying the steps to achieve them. It includes languages such as Prolog, SQL, and XSLT.

Model (State)

View

$$f \left( \begin{array}{l} \text{name: John} \\ \text{surname: Dough} \end{array} \right) =$$

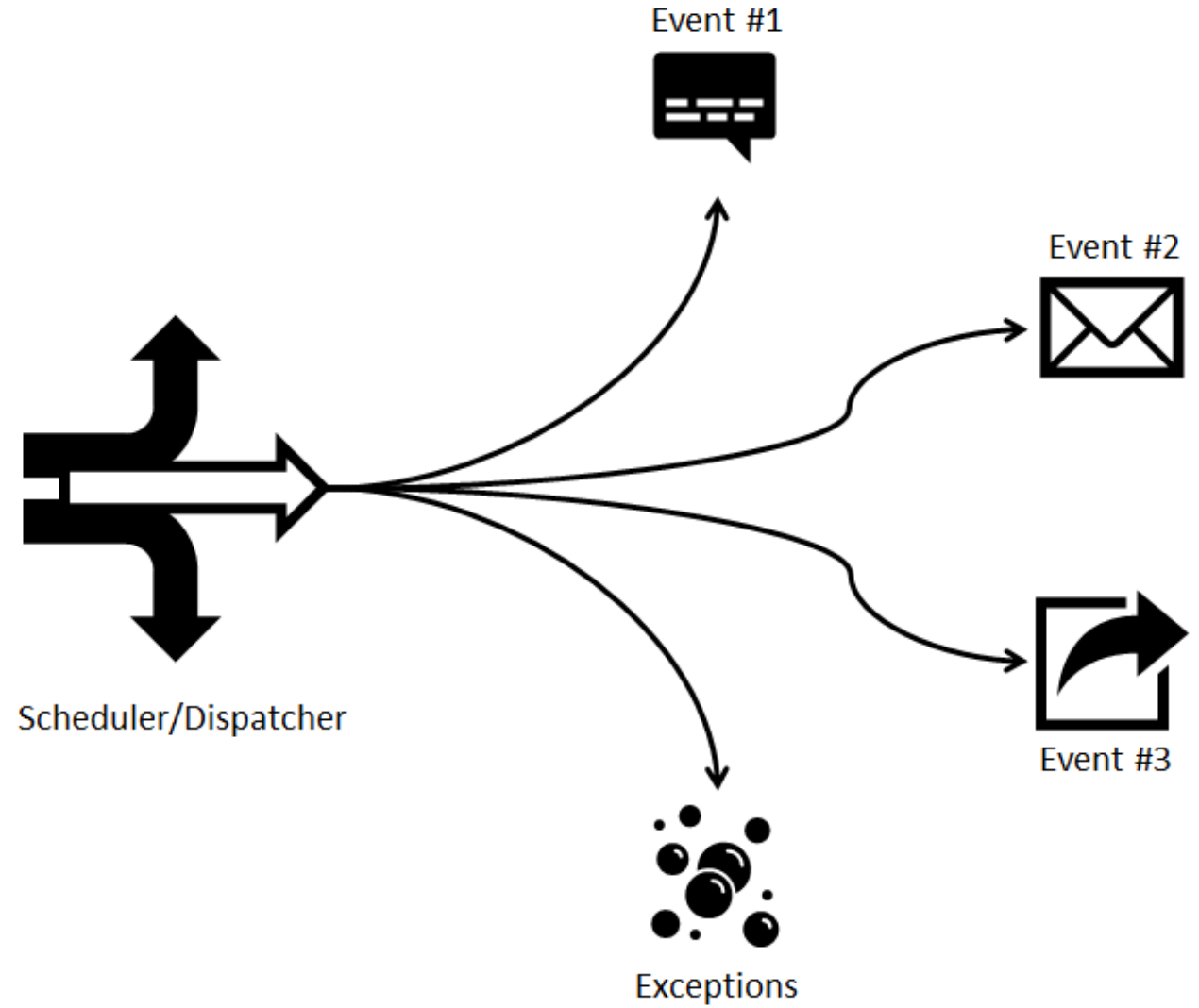
John

Dough

OK

# Event-Driven Programming

This type of programming is based on the reaction to events, such as user interactions or incoming data. It includes languages such as JavaScript and Visual Basic for Applications.



This type of programming is typically used for small, quick tasks and automating repetitive procedures. It includes languages such as Perl, PHP, and Python.

This type of programming is typically used for small, quick tasks and automating repetitive procedures. It includes languages such as Perl, PHP, and Python.

```

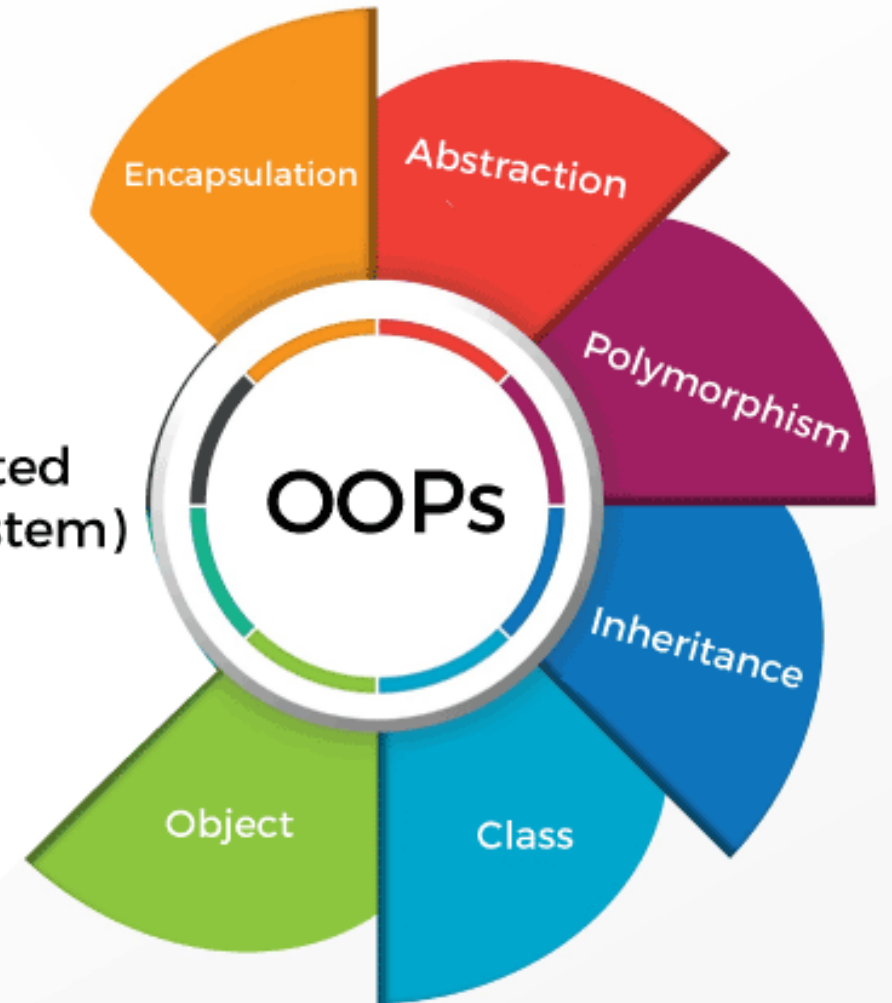
100 //total -> $total;
101 //taxes -> $taxes;
102 //total -> $total;
103 });
104
105 $old Taxes = $taxes;
106 $taxes = array();
107
108 $sort_order = array();
109
110 $results = $this->model->extension->getExtensions('total');
111
112 foreach ($results as $key => $value) {
113     if (isset($value['code'])) {
114         $code = $value['code'];
115     } else {
116         $code = $value['key'];
117     }
118     $sort_order[$key] = $this->config->get($code . '_sort_order');
119 }
120
121 array_multisort($sort_order, SORT_ASC, $results);
122
123 foreach ($results as $result) {
124     if (isset($result['code'])) {
125         $code = $result['code'];
126     } else {
127         $code = $result['key'];
128     }
129     if ($this->config->get($code . 'status') {
130         $this->load->model("extension/total/" . $code);
131
132         // we have to put the totals in an array so that they pass
133         // by reference.
134         $this->["model_extension_total_" . $code] => get_total($
135             total_data);
136
137         if (empty($totals[count($totals) - 1])) {
138             $totals[count($totals) - 1] = $code;
139         }
140     }
141 }
142
143 $tax_difference = 0;
144
145 foreach ($taxes as $tax_id => $value) {
146     if (isset($old Taxes[$tax_id])) {
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

# Object-Oriented Programming

This type of programming is based on the concept of objects and classes and focuses on encapsulating data and behavior into objects. It includes languages such as Java, Python, and C#.

( Object - Oriented Programming System)





# Outline

- Module Induction
- Background to module (the why)
- Introduction to Computer Programming
- **Basics of programming**
- Natural Language
- Translators
- Compilers
- Assemblers





# Basics of Programming

A set of fundamental concepts that are essential for understanding and writing computer programs.

These concepts form the foundation of computer programming and are used in almost all programming languages.

Understanding these concepts is crucial for writing efficient and effective programs.



# Basics of Programming

- Algorithms
- Variables
- Data Types
- Operators
- Conditional Statements
- Loops

# Basics of Programming

- **Algorithms**

- a set of well-defined steps used to solve a problem. Algorithms are the basis of all computer programs.

- **Variables**

- A variable is a named storage location used to hold data in a program. Variables have a data type, such as integer, string, or boolean, which determines the type of data that can be stored in the variable.

- **Data Types**

- A variable is a named storage location used to hold data in a program. Variables have a data type, such as integer, string, or boolean, which determines the type of data that can be stored in the variable.

- **Operators**

- Operators are symbols that perform operations on variables, such as addition, subtraction, multiplication, and division.

- **Conditional Statements**

- Conditional statements allow a program to make decisions based on the values of variables. They include if/else statements, switch statements, and ternary operators.

- **Loops**

- Loops allow a program to repeat a block of code multiple times, until a certain condition is met. Examples of loops include for loops, while loops, and do/while loops.



# Basics of Programming

- **Algorithms**
  - a set of well-defined steps used to solve a problem. Algorithms are the basis of all computer programs.



# Basics of Programming

- **Algorithms**

- a set of well-defined steps used to solve a problem. Algorithms are the basis of all computer programs.

- **Variables**

- A variable is a named storage location used to hold data in a program. Variables have a data type, such as integer, string, or boolean, which determines the type of data that can be stored in the variable.

# Basics of Programming

- **Algorithms**

- a set of well-defined steps used to solve a problem. Algorithms are the basis of all computer programs.

- **Variables**

- A variable is a named storage location used to hold data in a program. Variables have a data type, such as integer, string, or boolean, which determines the type of data that can be stored in the variable.

- **Data Types**

- A variable is a named storage location used to hold data in a program. Variables have a data type, such as integer, string, or boolean, which determines the type of data that can be stored in the variable.

# Basics of Programming

- **Algorithms**

- a set of well-defined steps used to solve a problem. Algorithms are the basis of all computer programs.

- **Variables**

- A variable is a named storage location used to hold data in a program. Variables have a data type, such as integer, string, or boolean, which determines the type of data that can be stored in the variable.

- **Data Types**

- A variable is a named storage location used to hold data in a program. Variables have a data type, such as integer, string, or boolean, which determines the type of data that can be stored in the variable.

- **Operators**

- Operators are symbols that perform operations on variables, such as addition, subtraction, multiplication, and division.

# Basics of Programming

- **Algorithms**

- a set of well-defined steps used to solve a problem. Algorithms are the basis of all computer programs.

- **Variables**

- A variable is a named storage location used to hold data in a program. Variables have a data type, such as integer, string, or boolean, which determines the type of data that can be stored in the variable.

- **Data Types**

- A variable is a named storage location used to hold data in a program. Variables have a data type, such as integer, string, or boolean, which determines the type of data that can be stored in the variable.

- **Operators**

- Operators are symbols that perform operations on variables, such as addition, subtraction, multiplication, and division.

- **Conditional Statements**

- Conditional statements allow a program to make decisions based on the values of variables. They include if/else statements, switch statements, and ternary operators.



# Basics of Programming

- **Algorithms**

- a set of well-defined steps used to solve a problem. Algorithms are the basis of all computer programs.

- **Variables**

- A variable is a named storage location used to hold data in a program. Variables have a data type, such as integer, string, or boolean, which determines the type of data that can be stored in the variable.

- **Data Types**

- A variable is a named storage location used to hold data in a program. Variables have a data type, such as integer, string, or boolean, which determines the type of data that can be stored in the variable.

- **Operators**

- Operators are symbols that perform operations on variables, such as addition, subtraction, multiplication, and division.

- **Conditional Statements**

- Conditional statements allow a program to make decisions based on the values of variables. They include if/else statements, switch statements, and ternary operators.

- **Loops**

- Loops allow a program to repeat a block of code multiple times, until a certain condition is met. Examples of loops include for loops, while loops, and do/while loops.



# Outline

- Module Induction
- Background to module (the why)
- Introduction to Computer Programming
- Basics of programming
- **Natural Language**
- Translators
- Compilers
- Assemblers

# Pause for Question



# Computer's Natural Language

- Formal, human-readable languages used to write instructions for computers to execute.
- Most common types are machine language, assembly language, high-level programming languages, markup languages, scripting languages, query languages, and domain-specific languages.

# Computer's Natural Language

- Computer's Natural Language is **Machine Language** or **Assembly Language**.
- Machine Language are low-level programming language that consists of binary codes (0s and 1s).
  - 0s and 1s directly represent instructions for the computer to perform.
- Assembly Language is also low-level language but uses mnemonics and symbols to represent machine language instructions. Providing a more readable
- Easy and efficient for the computer but difficult for humans to read and write.



# Computer's Natural Language

High-level programming languages: These are programming languages that use a syntax and structure closer to natural language, making it easier for humans to write, read, and understand code. Examples include **Python**, Java, and C++.



# Question

If computers only “understand” Machine Language, how then are they able to understand and process high-level languages, such as Python?



# Outline

- Module Induction
- Background to module (the why)
- Introduction to Computer Programming
- Basics of programming
- Natural Language
- **Translators**
- Compilers
- Assemblers



# Translators

- Translators are software programs that convert code written in one computer language into another language. There are several types of translators, including:
  - **Compilers:** These translators convert high-level programming code into machine language, which can then be executed directly by the computer's central processing unit (CPU).

# Translators

- Translators are software programs that convert code written in one computer language into another language. There are several types of translators, including:
  - **Interpreters:** These translators execute code written in a high-level language line by line, converting each line into machine language as it is executed.

# Translators

- Translators are software programs that convert code written in one computer language into another language. There are several types of translators, including:
  - **Assemblers**: These translators convert assembly language code into machine language.

# Translators

- Translators are software programs that convert code written in one computer language into another language. There are several types of translators, including:
  - **Linkers:** These translators combine object code generated by compilers into a single executable file.

# Translators

- Translators are software programs that convert code written in one computer language into another language. There are several types of translators, including:
  - **Loaders:** These translators load executable code into memory, where it can be executed by the CPU.



# Translators

- Translators play a crucial role in the software development process, allowing developers to write code in high-level languages that are easier to read, write, and understand, and then convert that code into machine language that can be executed by the computer.



# Outline

- Module Induction
- Background to module (the why)
- Introduction to Computer Programming
- Basics of programming
- Natural Language
- Translators
- **Compilers**
- Assemblers



# Compilers

- A translator that takes in high-level programming code and converts it into machine language, which can be executed directly by the computer's CPU.
- A compiler takes the source code written in a high-level language and translates it into an executable program that can run on a computer.
- The process of compiling involves several steps, including lexical analysis, parsing, semantic analysis, code generation, and optimization.





# Compilers

- Compilers play a crucial role in the software development process by allowing developers to write code in high-level languages that are easier to read, write, and understand, and then convert that code into machine language that can be executed by the computer.
- They also enable cross-platform development, allowing code written on one operating system to be compiled and run on another.



# Compilers

- Examples of popular compilers include GCC (GNU Compiler Collection) for C and C++, the Java compiler for Java, and the Microsoft C++ compiler for C++ on Windows.

# Python Compilers?

- For Python, the process of converting code into machine language is different than with traditional compiled languages like C or C++.
- Python is an interpreted language, which means that its code is executed line-by-line by an interpreter, rather than being compiled into machine language beforehand.
- The Python interpreter reads the source code and executes the instructions directly, allowing developers to write, test, and debug their code more quickly and easily than with compiled languages.
- This also makes Python a great language for rapid prototyping and experimentation.

# Python Trade-off

- The trade-off for this convenience is that interpreted code is generally slower than compiled code.
- Since the interpreter must translate the code into machine instructions each time it is executed.
- Some tools and techniques, such as pre-compiling the code into bytecode or using just-in-time (JIT) compilation, have been developed to improve the performance of interpreted Python code, but it still remains slower than compiled languages like C or C++.

# Hybrid Translators

- Hybrid translators are a type of translator that combine aspects of both compiled and interpreted languages.
- They provide the benefits of both models by translating the code into a more efficient intermediate representation, which is then executed by the computer.
- One example of a hybrid translator is a Just-In-Time (JIT) compiler. A JIT compiler compiles the code on-the-fly, as the program is executed, rather than compiling the entire program beforehand.
- The JIT compiler can optimize the code for the specific environment in which it is running, taking into account the available resources and system architecture.

# Hybrid Translators

- Another example of a hybrid translator is a pre-compiler, which converts the code into an intermediate representation that is faster to execute than the original source code.
- The pre-compiler can be used to speed up the startup time of an interpreted language by pre-compiling parts of the code that take a long time to execute.
- An example of a language that uses pre-compilation is Python. In Python, code can be pre-compiled into a binary format called "bytecode". The bytecode is stored in **.pyc** files, which can then be executed by the Python interpreter much faster than the original source code.

# Hybrid Translators

- By combining the advantages of both compiled and interpreted languages, hybrid translators provide a way to achieve the best of both worlds, with improved performance and ease of development.



# Outline

- Module Induction
- Background to module (the why)
- Introduction to Computer Programming
- Basics of programming
- Natural Language
- Translators
- Compilers
- **Assemblers**





# Assembler

- An assembler is a type of computer program that takes assembly language code as input and generates machine code as output.
- Assembly language is a low-level programming language that is used to write instructions for a computer's central processing unit (CPU).
- Assembly language code is more closely related to the machine code executed by the computer than high-level programming languages like Python or Java.



# Assembler

- Each instruction in assembly language corresponds directly to a single machine instruction, making it possible to write very efficient and tightly optimized code.
- However, assembly language code is often difficult to read and write, and can be prone to bugs and errors.
- The role of the assembler is to translate the assembly language code into machine code that can be executed by the computer.
- The assembler performs this translation by converting each assembly language instruction into the corresponding machine code instruction and resolving symbolic addresses into actual memory addresses.

# Assembler

- The use of assembly language is typically limited to system programming, embedded systems, and other specialized applications where the performance benefits of low-level programming are necessary.
- For most programming tasks, higher-level programming languages like Python, Java, or C are used instead, as they provide a more convenient and readable programming experience, while still offering good performance.



# Thank you

---

