# Recitation 1 for AF3214

Semester 2, 2024/25

The purpose of the recitations is to expand upon course materials covered in lecture and allow students to practice working with the material in an interactive setting.

# Agenda

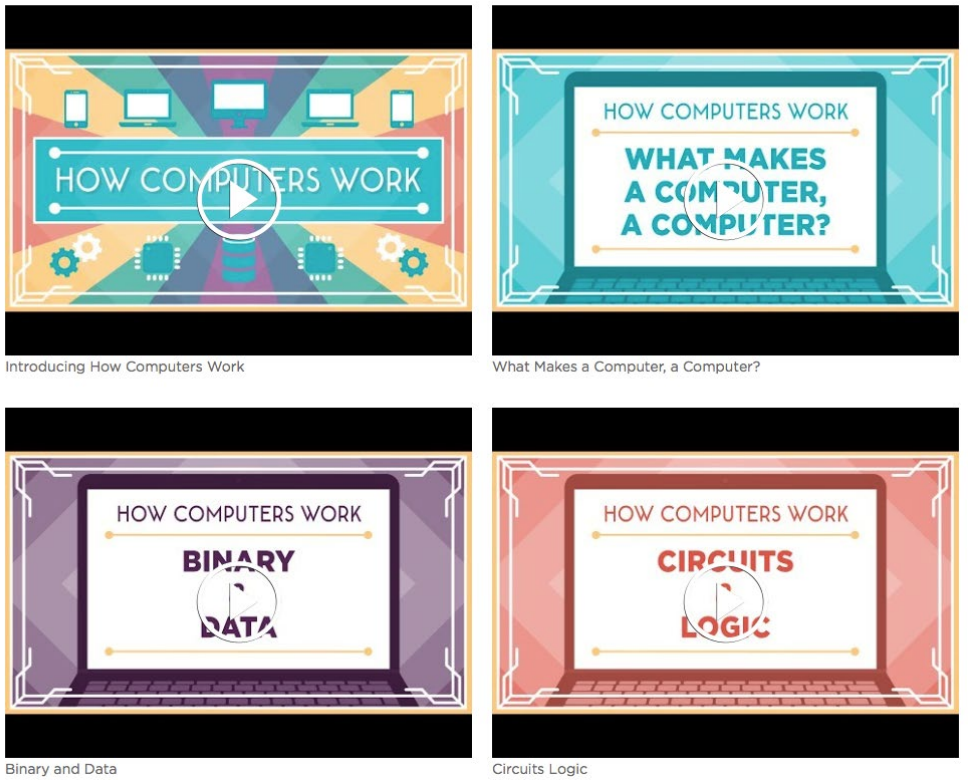
- Background
- Data types and variables
- Program flow
- Read and write
- Next level

# Programming knowledge

- No programming knowledge

- How computers work
  - Videos from code.org
  - https://code.org/educate/resources/videos

How Computers Work

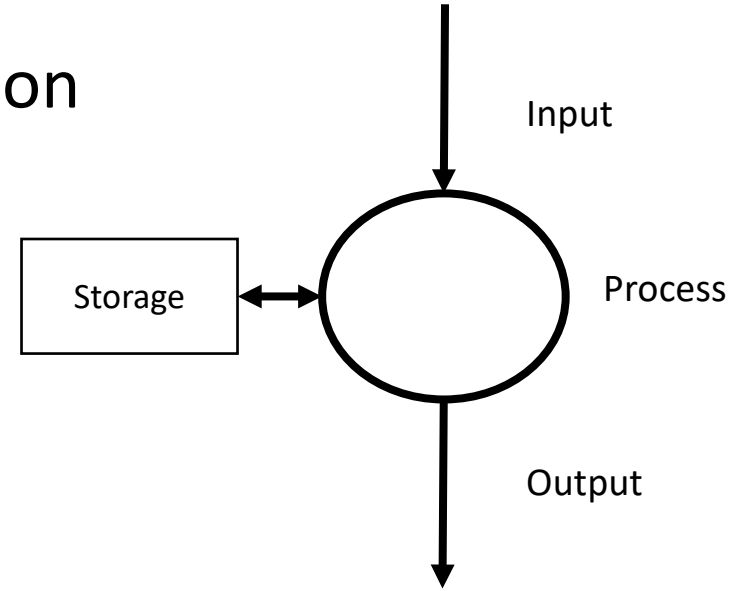



Introducing How Computers Work | What Makes a Computer, a Computer?

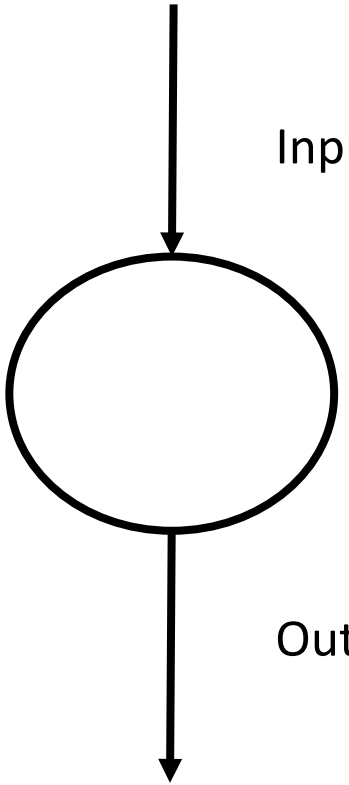
Binary and Data | Circuits Logic

# How are computers programmed?

- Explicit commands
- Thread of execution



# Writing code



Input keyboard, file

Process plain text files with extension py

Output screen, file

# Programming

- Programmers start from the narrative of problem they want to solve

- How is the problem going to be solved?
  - Input
  - Process
  - Output
  - Test data

# Data types

- Numbers
  - Integer
  - Decimal (float)
- String
- Booleans/binary
- Date/time (suggested next level)

- Common confusion: digits and number
  - A digit is any one of these symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. For example, the number 23 is written with two digits, 2 and 3.
  - A number is an amount of something. It can be written with one or more – or many – digits. Numbers can also be written with words. E.g., forty-seven

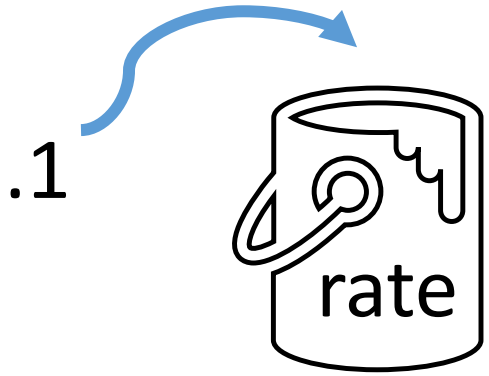
# Data type relevance

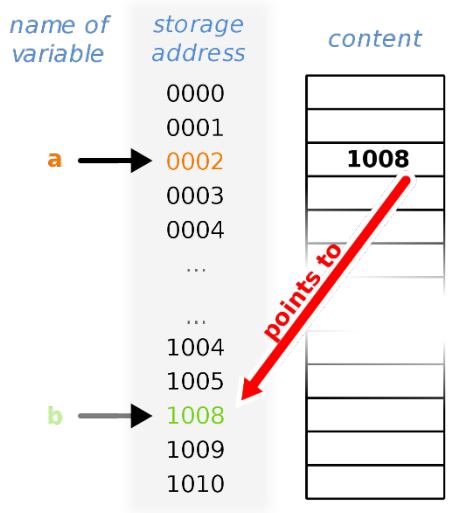
- Define valid data
- Define valid operations

- Adding two integers
  - 3 + 2 results in 5
- Adding two strings
  - “3”+”2” results in “32”

# Variables

- Change in value (similar concept to variables in algebra)
- Conceptually, like a bucket that stores values (technically, pointers to memory)



# Assigning values to variables

rate = .1

The variable name is always to the left of the equal sign

Python does not require to specify in advance the data type that the variable will store (declaring or dimensioning the variable).

Data type is inferred by the value stored.

Data type can be overwritten.

# Examples of assigning values to variables

| | |
|---|---|
| amount = 100 | ✓ |
| 100 = amount | ✗ |
| interest = amount * rate | ✓ |
| amount * rate = amount | ✗ |
| amount = “deductible” | ✓ |
| else = 100 | ✗ |

# Variable names

- Meaningful names
- Lowercase separated by underscore
- Python is case sensitive
  - Amount is not the same variable as amount
- Not reserved names

# Reserved words

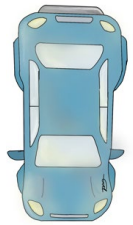
| | | | | |
|---|---|---|---|---|
| False | class | finally | is | return |
| None | continue | for | lambda | try |
| True | def | from | nonlocal | while |
| and | del | global | not | with |
| as | elif | if | or | yield |
| assert | else | import | pass | |
| break | except | in | raise | |

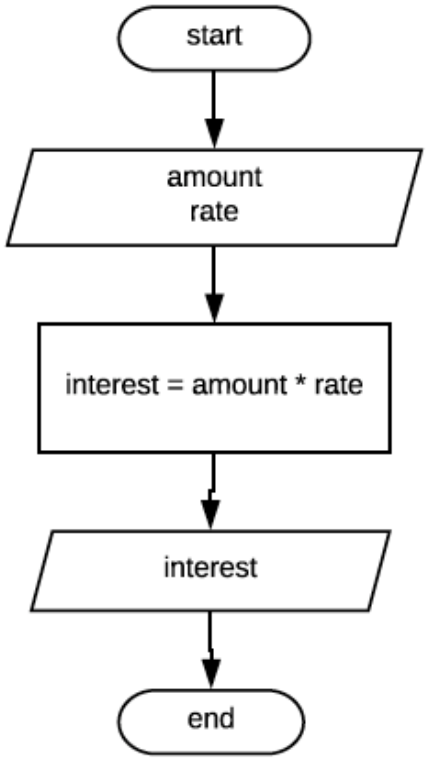
https://realpython.com/lessons/reserved-keywords/

# Program flow

- Path of execution
  - Unique (one and only one)
  - Divergent (if-then-else, make a judge, decide which way to go)
  - Loops (cycles)
    - Known number of repetitions (for)
    - Unknown number of repetitions (while)

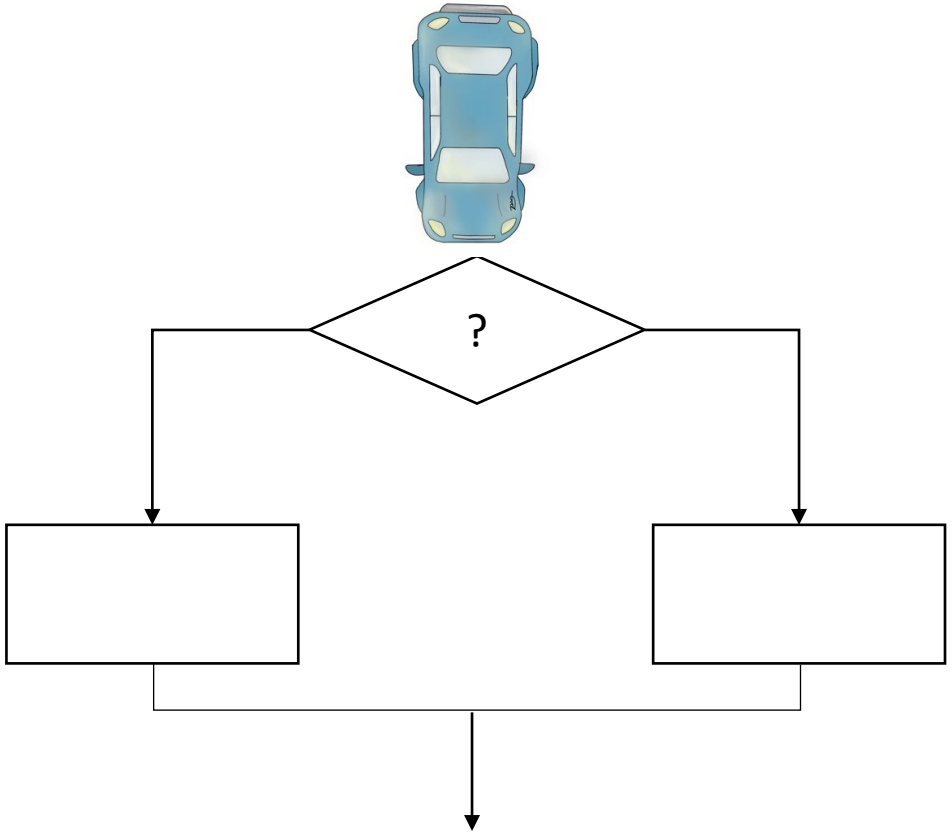
# Unique path of execution (Problem 1)






```
 9 # Input
10
11 amount = float(input ('Amount? '))
12 rate = float(input ('Interest rate? '))
13
14 # Process
15
16 interest = amount * rate
17
18 # output
19
20 print ('The annual interest is: ', interest)
21
```

# Divergent path




| Operators | Python |
|---|---|
| Greater than | > |
| Greater than or equal | >= |
| Less than | < |
| Less than or equal | <= |
| Equal | == |
| Different (not) | != |

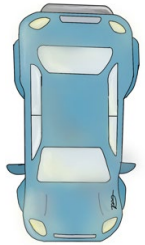
# Comparisons

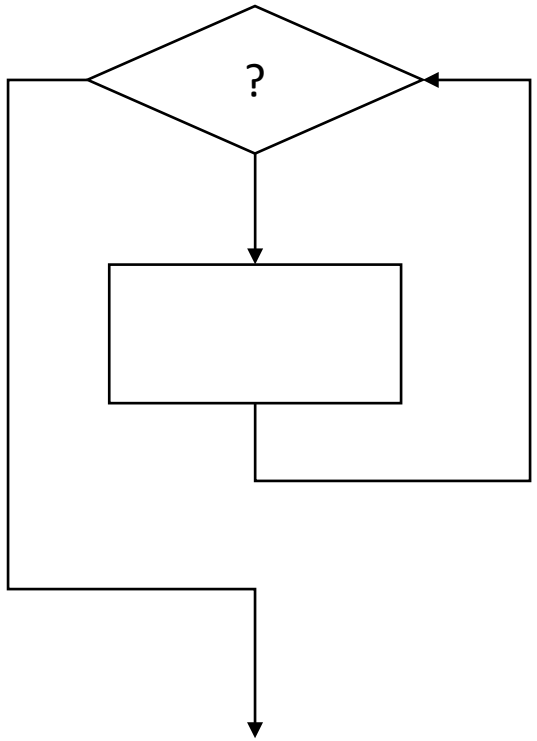
- Result in boolean values (True, False)
- Operators
- Logical operators for multiple simultaneous comparisons (and, or)

# Divergent path (if-then-else) (Problem 2)

```
 8 # Input
 9
10 amount = float(input ('Amount? '))
11 rate = float(input ('Interest rate? '))
12 interestType = input ('annual or monthly? ')
13 # Process
14 if interestType == 'annual':
15     interest = amount * rate
16 else:
17     interest = amount *rate / 12
18 # output
19
20 print ('The', interestType, 'interest is: ', interest)
```

# Loops






- Known number of repetitions (for)
- Unknown number of repetitions (while)

# Accumulators

- Initialization before the loop
- Update inside the loop
- Output usually outside the loop

```
total = total + amount
total += amount
```

# Loop with known number of repetitions (for)

- range (lower, upper)
  - range (1,3), cycles 2 times
  - range (3), cycles 3 times
- https://www.geeksforgeeks.org/python-range-function/

- for counter in range (1, 3)
- for i in range (1, limit+1)
- for i in range (1, j+1)

# Loop with known number of repetitions (for) (Problem 4)

```
10# Initializing accumulator
11
12total_interest=0
13
14# Input
15limit = int(input("How many notes receivable? "))
16
17for counter in range (limit):
18
19    # Input
20    amount = float(input ('Amount? '))
21    rate = float(input ('Interest rate? '))
22
23    # Process
24
25    interest = amount * rate
26
27    # Updating the accumulator
28    total_interest = total_interest + interest
29
30    # output
31
32    print ('The annual interest is: ', interest)
33
34
35
36print ('Total interest is: ',total_interest)
37
```

# Loop with unknown number of repetitions (while) (Problem 5)

```
10# Initialization of accumulator and flag
11
12total_interest=0
13flag = "y"
14
15while flag == "y":
16
17    #input
18    amount = float(input ('Amount? '))
19    rate = float(input ('Interest rate? '))
20
21    # Process
22
23    interest = amount * rate
24
25    # update accumulator
26    total_interest = total_interest + interest
27
28    # output
29
30    print ('The annual interest is: ', interest)
31
32    # input
33
34    flag = input("Another note receivable? yes[y] or no[n] ")
35
36
37print ('Total interest is: ',total_interest)
38
```

# Read and write

- File handlers
- Open/close files

# File paths in operating systems

- Windows: \ (back slash)
  - c:\Users\documents
- Mac and Linux: / (forward slash)
  - c:/users/documents
- Recommend: copy and paste

# Read and write modes

- “r” for reading (We will have an error if the file does not exist)

- “w” for writing only, which creates the file if it does not exist, and overwrites it if there is an existing file

# Read and write (Version 6)

```
10 # Initialization of accumulators
11
12 total_amount = 0
13 total_interest = 0
14 counter = 0
15
16 rate = float (input ('Rate? '))
17
18 amountFileHandler = open ("amount.txt",'r')
19
20 # interestFileHandler = open ("annual_interest.txt",'w')
21
22 print (amountFileHandler)
23
24
25 for line in amountFileHandler:
26     amount =float (line)
27     interest = amount * rate
28     print ('Amount: ', amount,', interest: ',interest)
29 #     interestFileHandler.write(str(interest) +'\n')
30     counter += 1
31     total_amount += amount
32     total_interest += interest
33
34 amountFileHandler.close()
35 # interestFileHandler.close()
36
37 print ('Total notes: ', counter)
38 print ('Total amount: ', total_amount)
39 print ('Total interest: ', total_interest)
40
```

# Python books

- Drawbacks of available books
  - Too technical
  - Not specific for accounting and finance
- Severance book and website (free). Python for everybody
  - https://www.py4e.com
- *How to Think Like a Computer Scientist-Learning with Python*, by Allen Downey, Jeff Elkner and Chris Meyers. Green Tea Press (https://greenteapress.com/wp/learning-with-python/)
- https://www.w3schools.com/python/default.asp
- Horstman and Necaise. Python for everyone. Wiley
- School library

# The End