TASK 1:-AI COMPANIES IN EGYPT

NO.	Company	Domain
1.	ABM Egypt	NLP, Social Media analytics, Search Engine optimization
2.	Adam.ai	NLP, machine learning, embedded systems
3.	Advansy ESC	Embedded Software, Big data, IoT
4.	Affectiva	Machine learning, Deep learning, Computer vision, Speech(new project(s))
5.	Agolo	NLP, ML, DL
6.	Arqamfc	Analytics
7.	Ave labs	Deep learning, Machine learning techniques, Computervision, Image Processing and Signal Processing, Control theory & Artificial Intelligence
8.	AvidBeam	Machine learning, OCR, Computer vision, DL
9.	Elves	Chatbots
10.	Bee Smart Payment Solutions	ML, Big Data
11.	Cequens	Predictive models, Machine Learning, Chatbots
12.	Cognitev	ML, DL, Recommendation systems, IR (Information Retrieval), NLP (Natural - Language Processing)
13.	Comiot	Iot, Computer vision, ML
14.	Crowd Analyzer	Big data, Machine learning, Social Media Analysis, Arabic in Social Media Analytics
15.	Data Gear BI	Data Mining, Business Intelligence(BI)
16.	El menus	Recommendation systems, NLP
17.	Heuro Labs	Artificial Intelligence, Cognitive Computing, Machine Learning, Big Data, Data Fusion and Data Mining
18.	IBM Egypt	NLP(Watson practice team), ML, DL
19.	Incorta	Robotics, computer vision
20.	InnoVisionSystems	Robotics, computer vision

21.	Itworx	Big Data
22.	Knowledgeofficer	information retrieval and text summarization for articleson the web. Based mainly for many NLP and recommendation systems
23.	Mendel.ai	Artificial Intelligence in Medicine, Machine learning,OCR,NLP
24.	MicrosoftATLc	ML, DL, NLP, OCR, Speech, Search engines(Bing team)
25.	Optomatica	Machine Learning, AI and Optimization solutions, Analytics
26.	Oracle	Machine learning, Big data
27.	Orange	ML, Big Data, Data Analytics
28.	Pixellion	Image processing, Machine learning, Deep learning, Computer vision, image recognition in the retail industry
29.	Raisa Energy	ML, predictive modeling
30.	RDI	Machine learning, Deep learning, Arabic/English OCR, Search engines, Information retrieval, Arabic/English NLPand Arabic/English speech recognition/identification
31.	Rology	Computer Vision, Medical Image Processing, Machine Learning
32.	Sadeem®	IoT & Business Analytics
33.	Seeloz	Data analytics, ML
34.	Soho SquareSolutions	IR (Information Retrieval), ML, NLP (Natural - LanguageProcessing)
35.	Speakol	ML, DL, Big Data
36.	Sypron	Big Data, ML, DL, Computer vision, NLP, General AI apps
37.	Stratochem	Data analysis, Image processing, data mining
38.	Tactful	Conversational Agents, NLU, DL, Information Retrieval
39.	Tahrir news	Machine learning, Deep learning, NLP(Arabic & English)
40.	TA Telecom	Big Data
41.	Teradata	Big Data Analytics, machine learning and data mining
42.	The D. GmbH	NLP, Chatbots, Computer Vision, Video Analysis
43.	Tpay	Machine learning

44.	Usy tech	NLP
45.	Valeo	ML, DL, reinforcement learning, Computer vision, Autonomous driving
46.	Wego	Data analysis, machine learning, recommendationsystems
47.	WideBot	Chatbots, NLP
48.	Wuzzuf	Machine learning, NLP(new project(s)), Search engines, Recommendation/Ranking systems and information retrieval
49.	Xtrava	Machine learning, Deep learning, Signal processing
50.	Yaoota!	Search engine, Big data, ML, NLP
51.	Vodafone	Big data, ML
52.	Wakeb Data	Business Intelligence, Data Analysis, Artificial Intelligence
53.	DilenyTech	Computer Vision
54.	SynapseAnalytics	ML, Computer Vision
55.	Asas Alqarar	Machine Learning, Deep Learning
56.	Dell EMC	Machine Learning, Big Data
57.	360imaging	Computer Guided Surgery, Medical Imaging
58.	Fruitful Solutions	Deep Learning, Computer Vision
59.	Advanced Intelligent Technologies	Machine Learning, Deep Learning
60.	Digified	Machine Learning, Computer Vision
61.	iHub	Machine Learning
62.	Smart MedicalServices	Machine Learning
63.	Okhtub	Machine Learning
64.	DevisionX	Deep Learning, Computer Vision
65.	ztechnium	Machine Learning, Recommendation Systems
66.	Lyra)	Machine Learning
67.	Cammedar Health	Machine Learning
68.	Botme	Chatbots, NLP
69.	Converted.in	Machine Learning
70.	MerQ	Chatbots, NLP

71.	Digisay	Machine Learning, Deep Learning
72.	Intouch	Machine Learning, Deep Learning
73.	Cassbana	Machine Learning, Deep Learning
74.	Pegasus	Data Analytics, BI
75.	Z2data	Data Analytics
76.	Dataplusme	Data Analytics, Data Science, Big Data Analytics, Data visualization
77.	3adda	Data Analytics, Business Intelligence
78.	Path Solutions	Big Data
79.	Ingram Micro	Data Science, Business Intelligence
80.	Scopic	Big Data
81.	TPay Mobile	Data Engineering, Data Analytics
82.	The Procter& Gamble	Data Science
83.	Henkel	Data Science, Business Intelligence
84.	Halan	Data Analysis, Business Intelligence, Data Science
85.	Mwasalat Misr	Data Analysis, Business Intelligence, Data Science
86.	Swvl	Data Analysis, Business Intelligence, Data Science
87.	Buseet	Data Analysis, Business Intelligence, Data Science
88.	Mawdoo3	Data Science, NLP
89.	Botitapp	Data Science, NLP
90.	Koinz	Data Analysis
91.	Brightskies	Machine Learning, Autonomous Driving
92.	DXWand	Machine Learning, text and speech analytics, NLP, analytics services
93.	MobiDev	AI apps
94.	Cyshield	Data Science
95.	Fixed Solutions	Data Science
96.	Nawy	Data Science, Business Intelligence
97.	Lenme	Machine Learning
98.	Atomica	Machine Learning, Computer Vision
99.	blnk.ai	Machine Learning
100.	Aimtechnolo	Data Science, Machine Learning

	gies	
101.	SURE InternationalTechnology	Machine Learning, AI
102.	BBI-Consultancy	Data Science
103.	Tensorgraph	AI, Voice Recognition
104.	Etisalat	Machine Learning, Data Science

TASK2:- TYPES OF ERROR IN PROGRAMMING

1. Syntax Errors

Just like human languages, computer languages have grammar rules. But while humans are able to communicate with less-than-perfect grammar, computers can't ignore mistakes, i.e. syntax errors.

For example, let's say the correct syntax for printing something is `print('hello')`, and we accidentally forget one of the parentheses while coding. A syntax error will happen, and this will stop the program from running.

Type this shortcut below

As your proficiency with programming language increases, you will make syntax errors less frequently. The easiest way to prevent them from causing you problems is to be aware of them early. Many text editors or IDEs will come with the ability to warn you about syntax errors at the time of writing.

2. Logic Errors

Logic errors can be the hardest to track down. Everything looks like it is working; you have just programmed the computer to do the wrong thing. Technically the program is correct, but the results won't be what you expected.

If you didn't check the requirements beforehand and wrote code to return the oldest user in your system when you needed the newest, you would have a logic error.

A famous example happened in 1999 when NASA lost a spacecraft due to miscalculations between English and American units. The software was coded one way but needed to work another.

When writing your tests, show them to the product manager or product owner to confirm that the logic you're about to write is correct. In the example above, someone closer to the business would have spotted that you aren't mentioning the fact it is the newest user that is required.

3. Compilation Errors

Some programming languages require a compilation step. Compilation is where your high-level language converts into a lower-level language that the computer can understand better. A compilation or compile-time error happens when the compiler doesn't know how to turn your code into the lower-level code.

In our syntax error example, if we were compiling `print('hello'`, the compiler would stop and tell us it doesn't know how to convert this into a lower-level language because it expected a `)` after the `'`.

If there is a compile-time error in your software, you won't be able to get it tested or launched.

Like syntax errors, you will get better at avoiding these with time, but in general, the best thing you can do is get early feedback when it happens.

Compilation happens across all files of your project at the same time. If you've made lots of changes and see lots of compiler warnings or errors, it can be very daunting. By running the compiler often, you will get the feedback you need sooner, and you will more easily know where to address the issues.

4. Runtime Errors

Runtime errors happen as a user is executing your program. The code might work correctly on your machine, but on the webserver, there might be a different configuration, or it might be interacted with in a way that could cause a runtime error.

If your system took the input from a form and tried to capitalize the first letter of a name by doing something like

```
params[:first_name].
```

`capitalize`, this would break if the form was sent without a first name.

Runtime errors are particularly annoying because they directly impact your end user. A lot of these other errors will happen when you're at your computer working on the code. These errors occur when the system is running and can stop someone from doing what they need to do.

Make sure you have good error reporting in place to capture any runtime errors and automatically open up new bugs in your ticketing system Try and learn from each bug report so that in future you can guard against this type of error.

Making use of frameworks and community-maintained code is an excellent way of minimizing these types of errors because the code is in many different projects, so it will have already encountered and fixed many issues.

5. Arithmetic Errors

An arithmetic error is a type of logic error but involves mathematics. A typical example when performing a division equation is that you cannot divide by zero without causing an issue. Very few people would write 5 / 0, but you might not think that the size of something in your system might sometimes be zero, which would lead to this type of error.

`ages.max / ages.min` could return an error if either `ages.max` or `ages.min` were zero.

Arithmetic errors can generate logic errors as we've discussed, or even run-time errors in the case of divide by zero.

Having functional tests that always include edge-cases like zero, or negative numbers is an excellent way to stop these arithmetic errors in their tracks.

6. Resource Errors

The computer that your program is on will allocate a fixed amount of resources to the running of it. If something in your code forces the computer to try and allocate more resources than it has, it can create a resource error.

If you accidentally wrote a loop that your code could never exit from, you would eventually run out of resources. In this example, the while loop will keep on adding new elements to an array. Eventually, you will run out of memory.

```
while(true)
  my_array << 'new array element'
end
```

Resource errors can be hard to chase down because the machine you're developing on can often be higher quality than the servers running your code. It is also hard to mimic real-world use from your local computer.

Having good reporting on resource usage on your web servers will flag code that is consuming too much of any type of resource over time.

Resource errors are an example of a type of error in programming that might be something for the operations team to fix rather than developers.

There are lots of load-testing applications and services that you can use to test what happens when multiple people try and run your code at once. Then, you can tune the testing to suit what is realistic for your application.

7. Interface Errors

Interface errors occur when there is a disconnect between how you meant your program to be used and how it is actually used. Most things in software follow standards. If input your program receives doesn't conform to the standards, you might get an interface error.

For example, an interface error might happen if you have an API that requires that specific parameters are set and those parameters are not set.

Unless handled correctly, interface errors will look like an error on your side when it is an error on the caller's side. This can lead to frustration from both sides.

Having clear documentation and catching these errors to pass back to the caller in a useful way is the best way of saying, "Hey, you haven't given us what we need to process this request." This will help keep support costs down and keep your clients happy because they know what they need to fix.

If you don't catch these errors and pass them back to the caller, they will end up appearing like runtime errors in your reporting, and you will end up guarding against things excessively.

TASK 3:-IS PHP SCRIPTING LANGUE ? OR NOT ?

Yes, PHP is a scripting language used mainly for server-side web development. Because of its open-source nature, PHP is a general-purpose language often used for other projects and graphical user interfaces.

TASK 4:-Example of interpreted langue

Ruby , Python, PHP ,JavaScript

TASK 5:- A programming paradigm and the best one

- Imperative: Programming with an explicit sequence of commands that update state.
- Declarative: Programming by specifying the result you want, not how to get it.
- Structured: Programming with clean, goto-free, nested control structures.
- Procedural: Imperative programming with procedure calls.
- Functional (Applicative): Programming with function calls that avoid any global state.
- Function-Level (Combinator): Programming with no variables at all.
- Object-Oriented Programming: by defining objects that send messages to each other. Objects have their own internal (encapsulated) state and public interfaces. Object orientation can be:
    - o Class-based: Objects get state and behavior based on membership in a class.
    - o Prototype-based: Objects get behavior from a prototype object.
- Event-Driven: Programming with emitters and listeners of asynchronous actions.

- Flow-Driven: Programming processes communicating with each other over predefined channels.
- Logic (Rule-based): Programming by specifying a set of facts and rules. An engine infers the answers to questions.
- Constraint: Programming by specifying a set of constraints. An engine finds the values that meet the constraints.
- Aspect-Oriented: Programming cross-cutting concerns applied transparently.
- Reflective: Programming by manipulating the program elements themselves.
- Array: Programming with powerful array operators that usually make loops unnecessary

→I think the best one it oop but not the fast