සියලු ම හිමිකම් ඇවිරිණි / *All Rights Reserved*]

Bandaranayake - ictfromabc Ravindu Bandaranayake - ictfromabc Ravindu Bandaranayake - ictfromabc Ravindu Bandaranayake - ictfromabc Ravindu Bandaranayake - ictfromabc Ravindu

Information and Communication Technology

තොරතුරු හා සන්නිවේදන තාක්ෂණය

2026 QUIZ NO 230 MARKING

2025/07/28
ictfromabc
Ravindu Bandaranayake

1. What is the output of the following code?
පහත කේතයේ ප්‍රතිදානය කුමක්ද?

```
nums = [10, 20]
temp = nums
nums.append([30, 40])
temp.remove(10)
nums.extend([50])
temp.clear()
nums.append(60)
print(nums)
```

1) `[20, [30, 40], 50, 60]`
2) `[60]`
3) `[[30, 40], 50, 60]`
4) `[10, 20, [30, 40], 50, 60]`
5) `[50,60]`

Answer : 2)

Explanation/ **පැහැදිලි කිරීම**

The list nums is first defined as [10, 20], and temp is set to reference the same list (not a copy). So changes to either nums or temp affect the same underlying list.
ලැයිස්තු අංක මුලින්ම [10, 20], **ලෙස අර්ථ දක්වා ඇති අතර, තාවකාලිකය එකම ලැයිස්තුවට යොමු වීමට සකසා ඇත (පිටපතක් නොවේ). එබැවින් සංඛ්‍යා හෝ තාවකාලික වෙනස්කම් එකම යටින් පවතින ලැයිස්තුවට බලපායි.**

NOTE
In Python, the = operator assigns a reference of a list (or any object) to another variable, meaning both variables point to the same object in memory.
Pythonහි, = **මෙහෙයවනය ලැයිස්තුවක (හෝ ඕනෑම වස්තුවක) යොමුවක් වෙනත් විචල්‍යයකට පවරයි, එනම් විචල්‍ය දෙකම මතකයේ ඇති එකම වස්තුවකට යොමු කරයි.**

So, changing one affects the other. On the other hand, the **`.copy()`** method creates a shallow copy of the original list.
ඉතින්, එකක් වෙනස් කිරීම අනෙකට බලපායි. අනෙක් අතට, **`.copy()`** **ක්‍රමය මුල් ලැයිස්තුවේ නොගැඹුරු පිටපතක් නිර්මාණය කරයි.**

This means the new list has the same elements, but it is stored at a different memory location, so changes to one do not affect the other
මෙයින් අදහස් කරන්නේ නව ලැයිස්තුවේ එකම අංග ඇති නමුත් එය වෙනත් මතක ස්ථානයක ගබඩා කර ඇති බැවින් එකකට කරන වෙනස්කම් අනෙකට බලපාන්නේ නැති බවයි

When we do `append([30, 40])`, it adds the whole list `[30, 40]` as one item, not separate numbers.
අපි `append([30, 40])`, **කරන විට, එය සම්පූර්ණ ලැයිස්තුව** `[30, 40]` **එක් අයිතමයක් ලෙස එකතු කරයි, වෙනම සංඛ්‍යා ලෙස නොවේ.**

```
nums=[ 10, 20, ,[30,40]]
```

Then `remove(10)` takes out the number 10. Next, `extend([50])` adds 50 to the end of the list.
ඉන්පසු `remove(10)` මඟින් 10 අංකය ඉවත් කරයි. ඊළඟට, `extend([50])` මඟින් ලැයිස්තුවේ අවසානයට 50 ක් එකතු කරයි.

```
nums=[20 ,[30,40],50 ]
```

When we call `clear()`, it empties the whole list because temp and nums are the same.
අපි `clear()`, ලෙස ඇමතූ විට, එය මුළු ලැයිස්තුවම හිස් කරයි, මන්ද temp සහ nums සමාන වේ.

```
nums=[]
temp=[]
```

Finally, `append(60)` adds the number 60 to the empty list. So, the list becomes [60] in the end.
අවසානයේ, `append(60)` හිස් ලැයිස්තුවට 60 අංකය එකතු කරයි. එබැවින්, ලැයිස්තුව අවසානයේ [60] බවට පත්වේ.

```
nums=[60]
```

Therefore, the answer is 2) /එබැවින් පිළිතුර 2) වේ

2. Which historical computing system is considered the first electronic general-purpose digital computer?
පළමු ඉලෙක්ට්‍රොනික පොදු කාර්ය ඩිජිටල් පරිගණකය ලෙස සැලකෙන පරිගණක පද්ධතිය කුමක්ද?

1) ENIAC 2) UNIVAC I 3) MARK 1 4) UNIVAC 5) IBM 701

Answer : 1)

Explanation/ පැහැදිලි කිරීම

This question is asking which early computer was the first ever to be fully electronic, digital, and capable of solving general-purpose problems (not just one specific task). That means it could be reprogrammed and used for many kinds of calculations.
මෙම ප්‍රශ්නය අසන්නේ කුමන මුල් පරිගණකයද යන්නයි. එය සම්පූර්ණයෙන්ම ඉලෙක්ට්‍රොනික, ඩිජිටල් සහ සාමාන්‍ය කාර්ය ගැටළු විසඳීමට හැකියාව ඇති පළමු පරිගණකය කුමක්ද යන්නයි (එක් නිශ්චිත කාර්යයක් පමණක් නොවේ). එයින් අදහස් කරන්නේ එය නැවත ක්‍රමලේඛනය කර බොහෝ ආකාරයේ ගණනය කිරීම් සඳහා භාවිතා කළ හැකි බවයි.

Let's see why the other options are incorrect./ අනෙක් විකල්ප වැරදි ඇයි සලකා බලමු.

Option 1
ENIAC (Electronic Numerical Integrator and Computer), built in 1945 in the U.S., was the **first fully electronic general-purpose digital computer**. It could be programmed to solve a wide variety of problems and used thousands of vacuum tubes for processing. This marked the beginning of the electronic computing era.
1945 දී ඇමරිකා එක්සත් ජනපදයේ ඉදිකරන ලද ENIAC ඉලෙක්ට්‍රොනික සංඛ්‍යාත්මක අනුකලකය සහ පරිගණකය, පළමු පූර්ණ ඉලෙක්ට්‍රොනික පොදු කාර්ය ඩිජිටල් පරිගණකය විය. එය විවිධ ගැටළු විසඳීම සඳහා වැඩසටහන්ගත කළ හැකි අතර සැකසුම් සඳහා ටික්ත නල දහස් ගණනක් භාවිතා කළේය. මෙය ඉලෙක්ට්‍රොනික පරිගණක යුගයේ ආරම්භය සනිටුහන් කළේය.

Option 2
UNIVAC I (Universal Automatic Computer I), developed in 1951, was the first commercial computer sold in the U.S. It was used for business and government applications, but it came after ENIAC and was not the first of its kind.
1951 දී සංවර්ධනය කරන ලද UNIVAC I (Universal Automatic Computer I)එක්සත් ජනපදයේ අලෙවි කරන ලද පළමු වාණිජ පරිගණකය විය. එය ව්‍යාපාර සහ රජයේ යෙදුම් සඳහා භාවිතා කරන ලදී, නමුත් එය ENIAC පසුව පැමිණි අතර එය එවැනි ආකාරයේ පළමු පරිගණකය නොවේ.

Option 3
The Harvard Mark I, completed in 1944, was one of the earliest electromechanical computers. It used mechanical relays and switches, not fully electronic parts, so it was slower and more limited compared to ENIAC.
1944 දී නිම කරන ලද Harvard Mark I පැරනිතම විද්‍යුත් යාන්ත්‍රික පරිගණක වලින් එකකි. එය සම්පූර්ණයෙන්ම ඉලෙක්ට්‍රොනික කොටස් නොව යාන්ත්‍රික රිලේ සහ ස්විච භාවිතා කළ බැවින් එය ENIAC. හා සසඳන විට මන්දගාමී සහ සීමිත විය.

Option 4
This is just a shortened name for the UNIVAC I series. Like option 2, it refers to commercial use, not the original, fully electronic, general-purpose system.
මෙය UNIVAC I series සඳහා කෙටි නමක් පමනි. විකල්ප 2 මෙන්, එය මුල්, සම්පූර්ණයෙන්ම ඉලෙක්ට්‍රොනික, පොදු කාර්ය පද්ධතියට නොව වානිජමය භාවිතයට යොමු කරයි.

Option 5
The IBM 701 was introduced in the early 1950s as IBM's first scientific computer. It was significant for the development of computing, but it came after both ENIAC and UNIVAC, and was not the first of its kind.
IBM 701 පරිගණකය 1950 ගණන්වල මුල් භාගයේදී IBM හි පළමු විද්‍යාත්මක පරිගණකය ලෙස හඳුන්වා දෙන ලදී. එය පරිගණක සංවර්ධනය සඳහා වැදගත් වූ නමුත් එය ENIAC සහ UNIVAC යන දෙකටම පසුව පැමිනි අතර එවැනි ආකාරයේ පළමු පරිගණකය නොවේ

Therefore the answer is 1) / එබැවින් පිළිතුර 1) වේ

3. What is the hexadecimal equivalent of the octal number 2364.56_8?
2364.56_8 අෂ්ටමය සංඛ්‍යාවේ ෂඩ් දශම සමානකම කුමක්ද?

1) $2D4.C3_{16}$ 2) $\underline{4F4.B8_{16}}$ 3) $3E7.A2_{16}$ 4) $5A1.9F_{16}$ 5) $4F3.C1_{16}$

Answer : 2)

Explanation/ පැහැදිලි කිරීම

We start by converting each digit of the octal number to a 3-digit binary number. Octal digits go from 0 to 7, and each can be written in 3 bits. So we take the number 2364.56_8 and break it down.
අපි ආරම්භ කරන්නේ අෂ්ටමය සංඛ්‍යාවේ සෑම ඉලක්කමක්ම ඉලක්කම් 3 ක ද්විමය සංඛ්‍යාවක් බවට පරිවර්තනය කිරීමෙනි. අෂ්ටමය සංඛ්‍යා 0 සිට 7 දක්වා වන අතර, ඒ සෑම එකක්ම බිටු 3 කින් ලිවිය හැකිය. එබැවින් අපි 2364.56_8 අංකය ගෙන එය බිඳ දමමු.

Base 8	2364.56_8					
	2	3	6	4	5	6
	010	011	110	100	101	110
Base 2	010011110100.101110					
finalized	10011110100.10111_2					

Next, we convert the binary number to hexadecimal. To do this, we group the binary digits in fours (4 bits) from left to right for the whole number part, and from right to left for the fraction part.
ඊළඟට, අපි ද්විමය සංඛ්‍යාව ෂඩ් දශම බවට පරිවර්තනය කරමු. මෙය සිදු කිරීම සඳහා, අපි ද්විමය සංඛ්‍යා හතරෙන් (බිටු 4) කාණ්ඩගත කරන්නේ සම්පූර්ණ සංඛ්‍යා කොටස සඳහා වමේ සිට දකුණට සහ භාග කොටස සඳහා දකුණේ සිට වමට ය.

Base 2	10011110100.10111_2				
Form	100	1111	0100	1011	1
	0100	1111	0100	1011	1000
	4	F	4	B	8
Base 16	$4F4.B8_{16}$				

Therefore, the answer is $4F4.B8_{16}$ which is the answer number 2)
එබැවින්, පිළිතුර $4F4.B8_{16}$ වන අතර එය පිළිතුරු අංකය 2) වේ

4. Calculate the sum of +45 and -25 using 8-bit 1's complement representation. What is the final result in binary?
බිටු-8 1හි අනුපූරක නිරූපණය භාවිතයෙන් +45 සහ -25 හි එකතුව ගණනය කරන්න. ද්විමයෙන් අවසාන ප්‍රතිඵලය කුමක්ද?

1) 00010100 2) 11101011 3) 01000110 4) 10111001 5) 00010011

Answer : 1)

Explanation/ **පැහැදිලි කිරීම**

- **Step 1** = Represent +45 and -25 in binary / **ද්විමය වශයෙන් +45 සහ -25 නියෝජනය කරන්න**

2	45	
2	22	- 1
2	11	- 0
2	5	- 1
2	2	- 1
2	1	- 0
	0	- 1

2	25	
2	12	- 1
2	6	- 0
2	3	- 0
2	1	- 1
	0	- 1

- **Step 2** = Convert to 1st complement / **පළමු අනුපූරකයට පරිවර්තනය කරන්න**

In 1's complement representation, only negative numbers are converted into their complement form. Positive numbers remain in their regular binary format.
1 හි අනුපූරක නිරූපණයේදී, සෘණ සංඛ්‍යා පමණක් ඒවායේ අනුපූරක ස්වරූපයට පරිවර්තනය වේ. ධන සංඛ්‍යා ඒවායේ සාමාන්‍ය ද්විමය ආකෘතියේ පවති.

So, to represent -25, we first write +25 in 8-bit binary:
එබැවින්, -25 නියෝජනය කිරීමට, අපි මුලින්ම +25 8-bit ද්විමය ආකාරයෙන් ලියන්නෙමු:
+25 = 00011001

Next, to find the 1's complement of this, we invert (flip) all bits, change every 0 to 1 and every 1 to 0:
ඊළඟට, මෙහි 1 හි අනුපූරකය සොයා ගැනීමට, අපි සියලු බිටු පෙරළිය යුතුය, සෑම 0 ක්ම 1 ට සහ සෑම 1 ක්ම 0 ට වෙනස් කරමු:
00011001 → 11100110

- **Step 3** = Perform the addition / **එකතු කිරීම සිදු කරන්න**

	0	0	1	0	1	1	0	1
+	1	1	1	0	0	1	1	0
1	0	0	0	1	0	0	1	1

This results in a 9-bit value. In 1's complement addition, we handle overflow by adding the 9th carry bit (the leftmost 1) back to the result.
මෙය බිටු-9 අගයක් ලබා දෙයි. 1 හි අනුපූරක එකතු කිරීමේදී, අපි 9 වන රැගෙන යාමේ බිටුව (වම් කෙළවර 1) ප්‍රතිඵලයට නැවත එකතු කළ යුතුය.

0	0	0	1	0	0	1	1
						+	1
0	0	0	1	0	1	0	0

Therefore the answer is 1) / **එබැවින් පිළිතුර** 1) **වේ.**

5. what is the most simplified expression of $(\overline{A+B})+\left(\overline{A+\overline{B}+C}\right)+\overline{A}B+\left(\overline{AB\overline{C}}\right)+C$ of the followings?
පහත සඳහන් ඒවායින් $(\overline{A+B})+\left(\overline{A+\overline{B}+C}\right)+\overline{A}B+\left(\overline{AB\overline{C}}\right)+C$ **හි වඩාත්ම සරල කළ ප්‍රකාශනය කුමක්ද?**

1) $A+B+C$ 2) $A+B$ 3) $A+\overline{B}+C$ 4) $A.B.C$ 5) $\underline{\overline{A}+\overline{B}+C}$

Answer : 5)

Explanation/ **පැහැදිලි කිරීම**

A Boolean expression is obtained and then simplified using Boolean laws
බූලියන් ප්‍රකාශනය ලබාගෙන පසුව බූලියන් නීති භාවිතයෙන් සරල කරනු ලැබේ

$F=(\overline{A+B})+\left(\overline{A+\overline{B}+C}\right)+\overline{A}B+\left(\overline{AB\overline{C}}\right)+C$

$=\overline{A}.\overline{B}+\left(\overline{A+\overline{B}+C}\right)+\overline{A}B+\left(\overline{AB\overline{C}}\right)+C$ De morgan's Law/ඩී මෝගන් න්‍යාය

$=\overline{A}.\overline{B}+\overline{A}.\overline{\overline{B}}.\overline{C}+\overline{A}B+\overline{A.B.\overline{C}}+C$ De morgan's Law [$(\overline{A+B})=\overline{A}.\overline{B}$ / ඩී මෝගන් න්‍යාය

$=\overline{A}.\overline{B}+\overline{A}.B.\overline{C}+\overline{A}B+\overline{A.B.\overline{C}}+C$ Double complement [$\overline{\overline{A}}=A$] / ද්විත්ව ප්‍රතිලෝම න්‍යාය

$=\overline{A}.\overline{B}+\overline{A}.B.\overline{C}+\overline{A}B+\overline{A}+\overline{B}+\overline{\overline{C}}+C$ De morgan's Law [$(\overline{A+B})=\overline{A}.\overline{B}$]/ ඩී මෝගන් න්‍යාය

$=\overline{A}.\overline{B}+\overline{A}.B.\overline{C}+\overline{A}B+\overline{A}+\overline{B}+C+C$ Double complement [$\overline{\overline{A}}=A$] ද්විත්ව ප්‍රතිලෝම න්‍යාය

$=\overline{A}.\overline{B}+\overline{A}.B.\overline{C}+\overline{A}B+\overline{A}+\overline{B}+C$ Idempotent Law [$A.A=A$]/ තදේවභාවී න්‍යාය

$=\overline{A}.\overline{B}+\overline{A}.B.\overline{C}+\overline{A}B+\overline{A}+\overline{B}+C$ Idempotent Law [$A.A=A$]/ තදේවභාවී න්‍යාය

$=\overline{A}.\overline{B}+\overline{A}.B.\overline{C}+\overline{A}B+\overline{A}+\overline{B}+C$ Idempotent Law [$A.A=A$]/ තදේවභාවී න්‍යාය

$=\overline{A}.\overline{B}+\overline{A}.B.\overline{C}+\overline{A}+\overline{B}+C$ Redundancy Law[$A+AB=A$]/සමතිරික්ත න්‍යාය

$=\overline{A}.\overline{B}+\overline{A}+\overline{B}+C$ Redundancy Law[$A+AB=A$]/සමතිරික්ත න්‍යාය

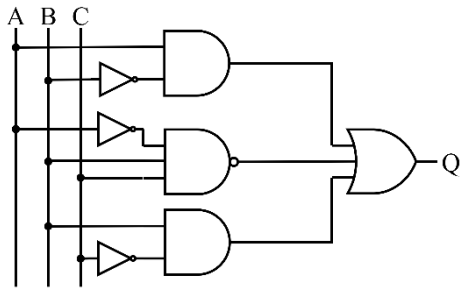
$=\overline{A}+\overline{B}+C$ Redundancy Law[$A+AB=A$]/සමතිරික්ත න්‍යාය

$=\overline{A}+\overline{B}+C$

$\overline{A}+\overline{B}+C$ is the most simplified Boolean expression./ **යනු වඩාත්ම සරල කළ බූලියන් ප්‍රකාශනයයි.**

Therefore, the answer is 5)/ **එබැවින් පිළිතුර** 5) **වේ**

6. What is the most simplified Boolean expression of the below logic circuit?
පහත තාර්කික පරිපථයේ සරලම බූලියන් ප්‍රකාශනය කුමක්ද?



1) $A+\overline{C}+\overline{AB}$
2) $A\overline{B}\overline{C}$
3) $AC+\overline{C}+\overline{A}\overline{B}$
4) $\overline{A}B+C$
5) $\underline{A+\overline{B}+\overline{C}}$

Answer : 5)

Explanation/ **පැහැදිලි කිරීම,**

- **Step 1** = Determine the Boolean expression from the circuit.
  පරිපථයෙන් බූලියන් ප්‍රකාශනය තීරණය කරන්න.

By following the connections and logic gates, the expression derived is:
සම්බන්ධතා සහ තාර්කික ද්වාර අනුගමනය කිරීමෙන්, ව්‍යුත්පන්න ප්‍රකාශනය වන්නේ:

$Q = A\overline{B} + \overline{\overline{A}BC} + B\overline{C}$

- **Step 2** = Let's simplify the above Boolean expression
  ඉහත බූලියන් ප්‍රකාශනය සරල කරමු.

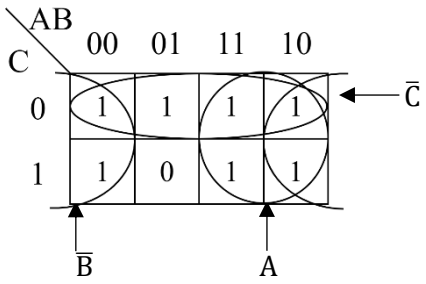
Method I – Using Boolean laws / **බූලියන් නීති භාවිතා කිරීම**

$A\overline{B} + \overline{\overline{A}BC} + B\overline{C}$ (De Morgans /)
$A\overline{B} + \overline{\overline{A}} + \overline{B} + \overline{C} + B\overline{C}$ (Double complement /)
$A\overline{B} + A + \overline{B} + \overline{C} + B\overline{C}$ (Redundancy / **සමතිරික්ත**)
$A + \overline{B} + \overline{C} + B\overline{C}$ (Redundancy / **සමතිරික්ත**)
$A + \overline{B} + \overline{C}$

Method II – Using Karnaugh map / **කානෝ සිතියම භාවිතා කිරීම**

A	B	C	$A\overline{B}$	$\overline{\overline{A}BC}$	$B\overline{C}$	Q
0	0	0	0	1	0	1
0	0	1	0	1	0	1
0	1	0	0	1	1	1
0	1	1	0	0	0	0
1	0	0	1	1	0	1
1	0	1	1	1	0	1
1	1	0	0	1	1	1
1	1	1	0	1	0	1

Using the values where the output Q = 1 in the truth table, we can now plot them on the Karnaugh map to simplify the Boolean expression.
සත්‍යතා වගුවේ Q = 1 ප්‍රතිදානය ඇති අගයන් භාවිතා කරමින්, බූලියන් ප්‍රකාශනය සරල කිරීම සඳහා අපට ඒවා කානෝ සිතියමේ සටහන් කළ හැකිය.



From the Karnaugh map, we group the adjacent 1s and get three groups. These groups simplify to $A, \overline{B}$ and $\overline{C}$, so the final simplified expression is $A + \overline{B} + \overline{C}$.
කානෝ සිතියමෙන්, 1හි ඒවා කාණ්ඩගත කර කණ්ඩායම් 3ක් ලබා ගත හැකිය. මෙම කණ්ඩායම් $A, \overline{B}$ සහ $\overline{C}$ ලෙස සරල කරයි, එබැවින් අවසාන සරල කළ ප්‍රකාශනය $A + \overline{B} + \overline{C}$ වේ.

Both methods above confirm that the simplified Boolean expression is $A + \overline{B} + \overline{C}$, and we can use either method to get the correct answer.
ඉහත ක්‍රම දෙකම සරල කළ බූලියන් ප්‍රකාශනය $A + \overline{B} + \overline{C}$ බව තහවුරු කරන අතර, නිවැරදි පිළිතුර ලබා ගැනීමට අපට ඕනෑම ක්‍රමයක් භාවිතා කළ හැකිය.

Therefore the answer is 5) / **එබැවින් පිළිතුර 5) වේ.**

7. What will be the most simplified form of $X + (X.Y + X) + \overline{X.Y}$?
$X + (X.Y + X) + \overline{X.Y}$ හි වඩාත් සරල කළ ප්‍රකාශය කුමක් වේද?

1) X+Y 2) X.Y 3) 0 4) 1 5) X

Answer : 4)

Explanation/ පැහැදිලි කිරීම

$= X + (XY + X) + \overline{XY}$
$= X + (XY + X) + \overline{X} + \overline{Y}$ De Morgan's Law (ද මුවාර් ප්‍රමේයය)
$= X + \overline{X} + (XY + X) + \overline{Y}$ Commutative Law (න්‍යාදේශ න්‍යාය)
$= 1 + (XY + X) + \overline{Y}$ Inverse Law (ප්‍රතිලෝම න්‍යාය)
$= 1$ Identity Law (සර්වසාම්‍ය න්‍යාය)

The correct answer here is 4) 1.
මෙහි නිවැරදි පිළිතුර වනුයේ 4) 1.

8. What will be the output of this code?
මෙම කේතයේ ප්‍රතිදානය කුමක් වේද?

```
count = 0
while count < 5:
    if count == 3:
        break
    count += 1
else:
    count = 10
print(count)
```

1) 3 2) 5 3) 10
4) Infinite loop 5) Error

Answer : 1)

Explanation/ පැහැදිලි කිරීම

- **`count = 0`**

Initializes the variable '`count`' to `0`. This will be used to control the while loop.
'`count`' විචල්‍යය `0` ට ආරම්භ කරයි. මෙය while ලූපය පාලනය කිරීමට භාවිතා කරනු ඇත.

- **`while count < 5:`**

Enters the loop as long as '`count`' is less than 5. Will repeat until the condition becomes false or a break occurs.
'`count`' 5 ට වඩා අඩු නම් ලූපයට ඇතුළු වේ. කොන්දේසිය අසත්‍ය වන තෙක් හෝ break සිදුවන තෙක් නැවත සිදු වේ.

- **`if count == 3:`**

Checks if the current value of '`count`' is 3. If so, it triggers the break to exit the loop.
'`count`' හි වත්මන් අගය 3 දැයි පරීක්ෂා කරයි. එසේ නම්, එය ලූපයෙන් පිටවීමට break ක්‍රියාත්මක කරයි.

- **`break`**

Immediately exits the loop if count is 3. Skips the rest of the loop and also avoids the else block.
count 3 නම් වහාම ලූපයෙන් පිටවෙයි. ඉතිරි ලූපය මඟ හරින අතර else බ්ලොක් එකද වළක්වයි.

- **`count += 1`**

If count is not 3, increment it by 1. This keeps the loop progressing toward the exit condition.
count 3 නොවේ නම්, එය 1 කින් වැඩි කරන්න. මෙය ලූපය පිටවීමේ තත්ත්වය දෙසට ඉදිරියට ගෙන යයි.

- **`else:`**

This '`else`' runs only if the loop wasn't exited by a `break`. If `break` occurs, this part is skipped.
මෙම '`else`' ක්‍රියාත්මක වන්නේ ලූපය `break` වලින් පිටව නොගියහොත් පමණි. `Break` සිදුවුවහොත්, මෙම කොටස මඟ හරිනු ලැබේ.

- **`count = 10`**

If loop ended normally (without `break`), set count to `10`. But in this case, this line will not be executed.
ලූපය සාමාන්‍යයෙන් අවසන් වුවහොත් (`break` නොමැතිව), ගණන් කිරීම `10` ලෙස සකසන්න. නමුත් මෙම අවස්ථාවේදී, මෙම රේඛාව ක්‍රියාත්මක නොවේ.

- **`print(count)`**

Prints the current value of '`count`'. Since the loop was broken at `count = 3`, output is: 3
'`count`' හි වත්මන් අගය මුද්‍රණය කරයි. loop එක `count = 3` හිදී කැඩී ගිය බැවින්, ප්‍රතිදානය: 3

Therefore the answer is 1) / එබැවින් පිළිතුර 1) වේ.

9. What is the output of the following Python code?
පහත පයිතන් කේතයේ ප්‍රතිදානය කුමක්ද?

```
for i in range(1, 6):
    print('{:<>6}'.format(i * '>'))
```

1) `<<<<<>`
`<<<<>>`
`<<<>>>`
`<<>>>>`
`<>>>>>`

2) `<<<<<>`
`<<<<>>`
`<<<>>>`
`<<>>>>`
`<<>>>>`

3) `<<<<<>`
`<<<<>>`
`<<>>>>`
`<<>>>>`
`<<>>>>`

4) `<<<<<>`
`<<<<>>`
`<<<>>>`
`<<>>>>`
`>>>>>>`

5) `<<<<<>`
`<<<<>>`
`<<<>>>`
`<<<<<<`
`<<>>>>`

Answer : 1)

Explanation/ පැහැදිලි කිරීම

The Python code runs a loop from **`i = 1`** to **`i = 5`**. For each iteration, it repeats the **`>`** symbol **`i`** times and uses the format **`{:<>6}`** to left-align this string within a 6-character space, filling the remaining space with `<`. As a result, the output is a series of lines that look like arrows increasing in size, with each line having the following output,
පයිතන් කේතය **`i = 1`** සිට **`i = 5`** දක්වා ලූපයක් ක්‍රියාත්මක කරයි. එක් එක් පුනරාවර්තනය සඳහා, එය **`>`** සංකේතය **`i`** වාරයක් iteration කරන අතර, මෙම **`string`** එක අකුරු 6ක ඉඩක් තුළ වමට පෙළගස්වා ගැනීමට, `<` සමඟ ඉතිරි ඉඩ පුරවා ගැනීමට **`{:<>6}`** ආකෘතිය භාවිතා කරයි. එහි ප්‍රතිඵලයක් වශයෙන්, ප්‍රතිදානය යනු ප්‍රමාණයෙන් වැඩි වන ඊතල මෙන් පෙනෙන රේඛා මාලාවකි, සෑම පේළියකටම පහත ප්‍රතිදානය ඇත,

```
<<<<<>
<<<<>>
<<<>>>
<<>>>>
<>>>>>
```

10. What is the output of the below Python code?
පහත පයිතන් කේතයේ ප්‍රතිදානය කුමක්ද?

```
x=0
while x<10:
    if x%2==0 or x%3==2:
        print(x, end='')
    x=x+1
```

1) 024568 2) 0124678 3) 0123456789 4) 013579 5) 02468

Answer : 1)

Let's consider the explanation of the above python code one by one.
ඉහත පයිතන් කේත රේඛාවේ පැහැදිලි කිරීම එකින් එක සලකා බලමු.

`x = 0`
Initially `x` is set to `0`.
මුලදී `x`, `0` ලෙස සකසා ඇත.

`while x < 10:`
The while loop will continue as long as `x` is less than `10`.
`x 10` ට වඩා අඩු වන තාක් ලූපය දිගටම පවතිනු ඇත.

Inside the loop, the if statement checks two conditions.
ලූපය තුළ, if ප්‍රකාශය කොන්දේසි දෙකක් පරීක්ෂා කරයි

- **`x % 2 == 0:`**

Checks if `x` is even (divisible by `2` with no remainder).
`x` ඉරට්ටේ දැයි පරීක්ෂා කරයි (ඉතිරි නොමැතිව `2` න් බෙදිය හැකිය).

- **`x % 3 == 2:`**

Checks if `x` gives a remainder of `2` when divided by `3`.
`3` න් බෙදූ විට `x`, ඉතිරියක් ලෙස `2` ලබා දෙන්නේ දැයි පරීක්ෂා කරයි.

If either condition is `True`, `x` is printed without a newline (due to `end=''`).
එක් කොන්දේසියක් වත් `True` නම්, `x` නව රේඛාවක් නොමැතිව මුද්‍රණය වේ (`end=''` නිසා).

`x` is incremented by `1` after each iteration.
එක් එක් පුනරාවර්තනයට පසුව `x`, `1` කින් වැඩි වේ.

x	x % 2 == 0	x % 3 == 2	Final Condition	Print
0	0 % 2 == 0 (True)	0 % 3 == 2 (False)	True	0
1	1 % 2 == 0 (False)	1 % 3 == 2 (False)	False	
2	2 % 2 == 0 (True)	2 % 3 == 2 (True)	True	2
3	3 % 2 == 0 (False)	3 % 3 == 2 (False)	False	
4	4 % 2 == 0 (True)	4 % 3 == 2 (False)	True	4
5	5 % 2 == 0 (False)	5 % 3 == 2 (True)	True	5
6	6 % 2 == 0 (True)	6 % 3 == 2 (False)	True	6
7	7 % 2 == 0 (False)	7 % 3 == 2 (False)	False	
8	8 % 2 == 0 (True)	8 % 3 == 2 (True)	True	8
9	9 % 2 == 0 (False)	9 % 3 == 2 (False)	False	

The code will output `024568`.
කේතය `024568` ලෙස ප්‍රතිදානය කරයි.

Each number is printed sequentially without spaces or newlines between them.
සෑම අංකයක්ම ඒවා අතර හිස් හෝ නව රේඛා නොමැතිව අනුපිළිවෙලින් මුද්‍රණය කරයි.

So, the correct answer is 1).
ඒ අනුව, නිවැරදි පිළිතුර 1) වේ.