

Information and Communication Technology

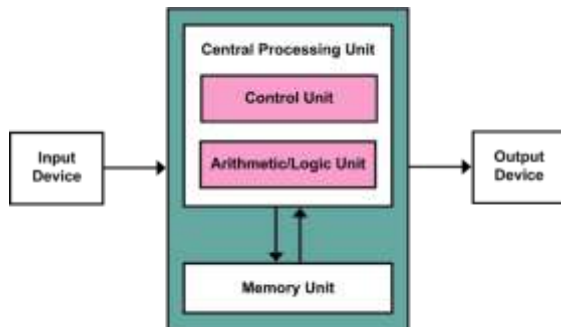
2025/10/02
ictfromabc
Ravindu Bandaranayake

- වැරදි වේ. මෙය සාප්තවම CPU මගින් නොව, ආදාන/ප්‍රතිදාන පාලකය (I/O controller) මගින් හසුරුවනු ලැබේ.

• **Option / විකල්පය 5)**

Incorrect. Long-term memory storage is managed by secondary storage devices like hard drives, not the CU.

වැරදි වේ. දිගු කාලීන මතක ගබඩාව කළමනාකරණය කරනු ලබන්නේ CU නොව, දෘඪ තැටි වැනි ද්විතියික ගබඩා උපාංග මගිනි



Therefore, the answer is, 3). / එබැවින් පිළිතුර 3) වේ.

2. Who developed the first, "Automatic Sequence Controlled Calculator" ?
පළමු "ස්වයංක්‍රීය අනුක්‍රමික පාලන ගණක යන්ත්‍රය" සංවර්ධනය කළේ කවුද?

- 1) Howard Aiken. / හෝවර්ඩ් අයිකන්.
- 2) Charles Babbage. / චාල්ස් බැබේජ්.
- 3) Blaise Pascal. / බ්ලේස් පැස්කල්.
- 4) Ada Augusta Lovelace. / ඒඩා ඔගස්ටා ලව්ලේස්.
- 5) Gottfried Wilhelm. / ගොට්ෆ්‍රිඩ් විල්හෙල්ම්.

Answer : 1)

1) **Howard Aiken / හොවර්ඩ් අයිකන්**

Howard Aiken, an American physicist and computer scientist, developed the first "Automatic Sequence Controlled Calculator," also known as the Harvard Mark I, in 1944. It was one of the earliest electromechanical computers and could perform complex calculations automatically based on sequences programmed on punched cards. This machine marked a significant advancement in computing technology.

ඇමරිකානු භෞතික විද්‍යාඥයෙකු සහ පරිගණක විද්‍යාඥයෙකු වන හොවර්ඩ් අයිකන් විසින් 1944 දී භාවයේ මාර්ක් I ලෙසද හැඳින්වෙන ප්‍රථම "ස්වයංක්‍රීය අනුක්‍රමික පාලන කැල්කියුලේටරය" නිපදවන ලදී. එය පැරණිතම විද්‍යුත් යන්ත්‍රික පරිගණක වලින් එකක් වූ අතර එය ක්‍රමලේඛනය කර ඇති අනුපිළිවෙල මත ස්වයංක්‍රීයව සංකීර්ණ ගණනය කිරීම් සිදු කළ හැකිය. පන්ච් කාඩ්. මෙම යන්ත්‍රය පරිගණක තාක්ෂණයේ සැලකිය යුතු දියුණුවක් සනිටුහන් කළේය.

2) **Charles Babbage / චාල්ස් බැබේජ්**

Charles Babbage is often referred to as the "father of the computer." He designed the Difference Engine and later conceptualized the Analytical Engine, which was a mechanical general-purpose computer. Although he never completed a working model, the Analytical Engine's design included many features found in modern computers, such as a central processing unit (CPU) and memory.

චාල්ස් බැබේජ් බොහෝ විට "පරිගණකයේ පියා" ලෙස හැඳින්වේ. ඔහු ඩිෆරන්ස් එන්ජින් නිර්මාණය කළ අතර පසුව යන්ත්‍රික පොදු කාර්ය පරිගණකයක් වූ විශ්ලේෂණ එන්ජින් සංකල්පගත කළේය. ඔහු කිසි විටෙක ක්‍රියාකාරී අකෘතියක් සම්පූර්ණ නොකළද, විශ්ලේෂණ එන්ජින් සැලසුමට මධ්‍යම සැකසුම් ඒකකය CPU සහ මතකය වැනි නවීන පරිගණකවල ඇති බොහෝ විශේෂාංග ඇතුළත් විය.

3) **Blaise Pascal / බ්ලේස් පැස්කල්**

Blaise Pascal was a French mathematician, physicist, and inventor. He developed the Pascaline, one of the first mechanical calculators, in 1642. The Pascaline could perform addition and subtraction directly and was used to help with tax calculations. It laid the groundwork for future developments in mechanical calculation devices.

බ්ලේස් පැස්කල් ප්‍රංශ ගණිතඥයෙක්, භෞතික විද්‍යාඥයෙක් සහ නව නිපැයුම්කරුවෙකු විය. ඔහු 1642 දී ප්‍රථම යන්ත්‍රික ගණක යන්ත්‍රවලින් එකක් වන Pascaline නිපදවීය. එය යන්ත්‍රික ගණනය කිරීමේ උපාංගවල අනාගත වර්ධනයන් සඳහා පදනම දැමීය.

4) Ada Augusta Lovelace

Ada Augusta Lovelace, often known simply as Ada Lovelace, was an English mathematician and writer. She is credited with writing the first algorithm intended to be processed by a machine, specifically Babbage's Analytical Engine. For this reason, she is considered the world's first computer programmer. Her work also included visionary ideas about the potential of computers to perform tasks beyond mere calculations.

Ada Augusta Lovelace බොනෝ විට සරලව Ada Lovelace ලෙස හැඳින්වෙන අතර, ඉංග්‍රීසි ගණිතඥයෙක් සහ ලේඛකයෙක් විය. යන්ත්‍රයකින්, විශේෂයෙන්ම Babbage's Analytical Engine මගින් සැකසීමට අදහස් කරන පළමු ඇල්ගොරිතම ලිවීමේ ගෞරවය ඇයට හිමි වේ. මේ හේතුව නිසා ඇය ලොව ප්‍රථම පරිගණක ක්‍රමලේඛිකාව ලෙස සැලකේ. හුදු ගණනය කිරීම්වලින් ඔබ්බට කාර්යයන් ඉටු කිරීමට පරිගණකවලට ඇති හැකියාව පිළිබඳ දුරදර්ශී අදහස් ද, ඇගේ කාර්යයට ඇතුළත් විය.

5) Gottfried Wilhelm Von

Gottfried Wilhelm von Leibniz was a German mathematician and philosopher. He developed the Stepped Reckoner, a mechanical calculator that could perform addition, subtraction, multiplication, and division. Leibniz is also known for his work in formal logic and for co-developing calculus independently of Isaac Newton.

Gottfried Wilhelm von Leibniz යනු ජර්මානු ගණිතඥයෙක් සහ දාර්ශනිකයෙක්. ඔහු එකතු කිරීම, අඩු කිරීම, ගුණ කිරීම සහ බෙදීම සිදු කළ හැකි යාන්ත්‍රික ගණක යන්ත්‍රයක් වන Stepped Reckoner නිර්මාණය කළේය. ලයිබ්නිස් ඔහුගේ විධිමත් තර්ක ශාස්ත්‍රය සහ අයිසැක් නිව්ටන්ගෙන් ස්වාධීනව කලනය සම-සංවර්ධනය කිරීම සඳහා ද ප්‍රසිද්ධය.

Therefore, the correct answer is 1) Howard Aiken.

එබැවින්, නිවැරදි පිළිතුර 1) හොවර්ඩ් අයිකන් වේ.

3. Which of the following statements about the hexadecimal number system is FALSE?

ඡබ්දගමය සංඛ්‍යා පද්ධතිය පිළිබඳ පහත සඳහන් කුමන ප්‍රකාශය අසත්‍යද?

- 1) Each hexadecimal digit represents a 4-bit binary sequence.
සෑම ඡබ්දගමය ඉලක්කමක්ම 4-bit ද්විමය අනුක්‍රමයක් මගින් නියෝජනය කරයි.
- 2) The hexadecimal system is commonly used in computer science and digital electronics.
පරිගණක විද්‍යාවේ සහ අංකිත ඉලෙක්ට්‍රොනික විද්‍යාවේ බහුලව භාවිතා වන්නේ ඡබ් දගමය පද්ධතියයි.
- 3) Hexadecimal numbers are base-16 numbers, meaning they use 16 distinct symbols.
ඡබ් දගමය සංඛ්‍යා යනු පාදය 16 වන සංඛ්‍යා වේ, එනම් ඒවා එකිනෙකට වෙනස් සංකේත 16ක් භාවිතා කරයි.
- 4) Hexadecimal numbers are often used to represent memory addresses and color codes.
මතක ලිපින සහ වර්ණ කේත නිරූපණය කිරීම සඳහා ඡබ්දගමය සංඛ්‍යා බොනෝ විට භාවිතා වේ.
- 5) Hexadecimal numbers can only represent positive integers.
ඡබ් දගම සංඛ්‍යා මගින් නියෝජනය කළ හැක්කේ ධන නිඛිල පමණි.

Answer : 5)

Each hexadecimal digit actually represents a 4-bit binary sequence, making this statement true. Therefore, it is not false.

සෑම ඡබ් දගම සංඛ්‍යාවක්ම ඇත්ත වශයෙන්ම 4-bit ද්විමය අනුක්‍රමයක් නියෝජනය කරයි, මෙම ප්‍රකාශය සත්‍ය කරයි. එබැවින් එය අසත්‍යයක් නොවේ.

The hexadecimal system is indeed commonly used in computer science and digital electronics. This statement is true.

පරිගණක විද්‍යාවේ සහ ඩිජිටල් ඉලෙක්ට්‍රොනික විද්‍යාවේ සැබවින් ම ඡබ් දගම පද්ධතිය බහුලව භාවිතා වේ. මෙම ප්‍රකාශය සත්‍යයි.

Hexadecimal numbers are indeed base-16 numbers, and they use 16 distinct symbols (0-9 and A-F) to represent values. This statement is also true.

ඡබ්දගමය සංඛ්‍යා ඇත්ත වශයෙන්ම පාද-16 සංඛ්‍යා වන අතර ඒවා අගයන් නිරූපණය කිරීමට වෙනස් සංකේත 16ක් 0-9 සහ A-F භාවිතා කරයි. මේ ප්‍රකාශයද සත්‍ය වේ.

Hexadecimal numbers are frequently used to represent memory addresses, color codes, and other data in computing systems. This statement is true.

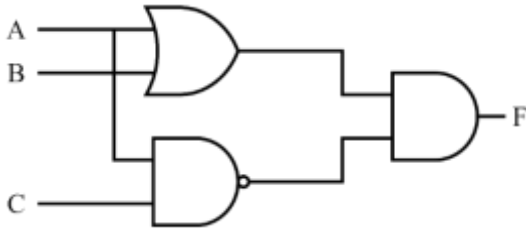
පරිගණක පද්ධතිවල මතක ලිපින, වර්ණ කේත සහ අනෙකුත් දත්ත නිරූපණය කිරීම සඳහා ඡබ්දගමය සංඛ්‍යා නිතර භාවිතා වේ. මෙම ප්‍රකාශයද සත්‍ය වේ.

Hexadecimal numbers can represent both positive and negative integers, just like decimal (base-10) numbers. The sign of the number is determined by the context in which it is used, such as in two's complement representation for signed integers in computer systems. Therefore, option 5 is false.

හෙක්සාදශමය සංඛ්‍යා දශම පාද-10 සංඛ්‍යා මෙන් ධන සහ ඍණ නිඛිල දෙකම නියෝජනය කළ හැක. සංඛ්‍යාවේ සලකුණ තීරණය වන්නේ පරිගණක පද්ධතිවල දක්වා ඇති පූර්ණ සංඛ්‍යා සඳහා දෙකේ අනුපූරක නිරූපණය වැනි එය භාවිතා කරන යම්කිසි සන්දර්භයක් මගිනි. එබැවින්, විකල්ප 5 අසත්‍ය වේ.

4. What is the correct SSOP expression corresponding to the logic circuit below?

පහත තාර්කික පරිපථයට අනුරූපණය වන නිවැරදි SSOP ප්‍රකාශනය කුමක්ද?



- 1) $(\overline{A}\overline{B}\overline{C}) + (\overline{A}\overline{B}C) + (\overline{A}B\overline{C}) + (\overline{A}BC)$
- 2) $(ABC) + (AB\overline{C}) + (\overline{A}BC) + (\overline{A}\overline{B}C)$
- 3) $(\overline{A}BC) + (\overline{A}\overline{B}C) + (AB\overline{C}) + (ABC)$
- 4) $(\overline{A}\overline{B}\overline{C}) + (\overline{A}\overline{B}C) + (\overline{A}B\overline{C}) + (\overline{A}BC)$
- 5) $(\overline{A}BC) + (AB\overline{C}) + (\overline{A}\overline{B}C) + (ABC)$

Answer : 4)

The Boolean expression corresponding to the above logic circuit is shown below.

ඉහත තාර්කික පරිපථයට අනුරූප වන බූලිය ප්‍රකාශනය පහතින් දැක්වේ.

$$(A + B). (\overline{A}. \overline{C})$$

There are 3 ways to convert that boolean expression to SSOP.

එම බූලිය ප්‍රකාශනය SSOP ආකාරයට පරිවර්තනය කිරීමට ආකාර 3ක් පවතී.

Method 1 :

Convert this statement to a SPOS expression and make it a SSOP.

මෙම ප්‍රකාශනය SPOS ප්‍රකාශනයක් බවට පත් කර එය SSOP බවට පත් කිරීම.

$$(A + B). (\overline{A}. \overline{C})$$

$$(A + B). (\overline{A} + \overline{C})$$

$$(A + B + 0). (\overline{A} + 0 + \overline{C})$$

$$(A + B + (C. \overline{C})). (\overline{A} + (B. \overline{B}) + \overline{C})$$

$$(A + B + C). (A + B + \overline{C}). (\overline{A} + B + \overline{C}). (\overline{A} + \overline{B} + \overline{C})$$

$$\text{SPOS} - (A + B + C). (A + B + \overline{C}). (\overline{A} + B + \overline{C}). (\overline{A} + \overline{B} + \overline{C})$$

$$(A + B + C). (A + B + \overline{C}). (\overline{A} + B + \overline{C}). (\overline{A} + \overline{B} + \overline{C})$$

$$(\overline{A} + \overline{B} + C). (\overline{A} + \overline{B} + \overline{C}). (\overline{A} + B + C). (\overline{A} + \overline{B} + C)$$

$$(\overline{A} + \overline{B} + C) + (\overline{A} + \overline{B} + \overline{C}) + (\overline{A} + B + C) + (\overline{A} + \overline{B} + C)$$

$$(\overline{A}\overline{B}C) + (\overline{A}\overline{B}\overline{C}) + (\overline{A}B\overline{C}) + (\overline{A}\overline{B}\overline{C})$$

$$(A\overline{B}C) + (A\overline{B}\overline{C}) + (A\overline{B}C) + (A\overline{B}\overline{C})$$

De Morgan's law ද, මූලාර්ථ ප්‍රමේයය

Identity law සර්වසාමය න්‍යාය

Inverse law ප්‍රතිලෝම න්‍යාය

Distributive law විභාජන න්‍යාය

Inverse of the above expression

ඉහත ප්‍රකාශනයේ ප්‍රතිලෝමය

De Morgan's law ද, මූලාර්ථ ප්‍රමේයය

De Morgan's law ද, මූලාර්ථ ප්‍රමේයය

Double inverse law ද්විත්ව ප්‍රතිලෝම

Method 2 :

Write down a truth table using the obtained Boolean expression to obtain an SSOP expression.

SSOP ප්‍රකාශනයක් ලබා ගැනීම සඳහා ලබාගත් බුලිය ප්‍රකාශනය භාවිතා කරමින් සත්‍යතා වගුවක් ලියන්න.

A	B	C	A + B	$\bar{A} \cdot \bar{C}$	$(A + B) \cdot (\bar{A} \cdot \bar{C})$
0	0	0	0	1	0
0	0	1	0	1	0
0	1	0	1	1	1
0	1	1	1	1	1
1	0	0	1	1	1
1	0	1	1	0	0
1	1	0	1	1	1
1	1	1	1	0	0

SSOP expression can also be derived using the minterms in the truth table above.

ඉහත සත්‍යතා වගුවේ minterms භාවිතා කරමින්ද SSOP ප්‍රකාශනයක් ව්‍යුත්පන්න කරගත හැක.

$$(\bar{A}\bar{B}\bar{C}) + (\bar{A}BC) + (A\bar{B}\bar{C}) + (AB\bar{C})$$

Method 3 :

Converting the above boolean expression into a SOP expression and converting it to the standard form.

ඉහත බුලියානු ප්‍රකාශනය SOP ප්‍රකාශනයක් බවට පත්කර එය සම්මත ආකාරයට පරිවර්තනය කර ගැනීම.

$$(A + B) \cdot (\bar{A} \cdot \bar{C})$$

$$(A + B) \cdot (\bar{A} + \bar{C})$$

$$A\bar{A} + A\bar{C} + B\bar{A} + B\bar{C}$$

$$0 + A\bar{C} + B\bar{A} + B\bar{C}$$

$$A\bar{C} + B\bar{A} + B\bar{C}$$

De Morgan's law ද මුළු ප්‍රමේයය

Distributive law විකටන න්‍යාය

Inverse law ප්‍රතිලෝම න්‍යාය

Identity law සර්වසාමය න්‍යාය

$$\text{SOP} - A\bar{C} + B\bar{A} + B\bar{C}$$

$$A1\bar{C} + B\bar{A}1 + 1B\bar{C}$$

$$A(B + \bar{B})\bar{C} + B\bar{A}(C + \bar{C}) + (A + \bar{A})B\bar{C}$$

$$AB\bar{C} + A\bar{B}\bar{C} + \bar{A}BC + \bar{A}B\bar{C} + ABC + \bar{A}B\bar{C}$$

$$AB\bar{C} + A\bar{B}\bar{C} + \bar{A}BC + \bar{A}B\bar{C}$$

Identity law සර්වසාමය න්‍යාය

Inverse law ප්‍රතිලෝම න්‍යාය

Distributive law විකටන න්‍යාය

Idempotent law තදේවතාවී න්‍යාය

5. Which of the following is NOT a characteristic of Karnaugh Maps (K-Maps)?

Karnaugh (K-Maps) හි ලක්ෂණයක් නොවන්නේ කුමක්ද?

- 1) K-Maps are used for simplifying Boolean expressions.
බුලියන් ප්‍රකාශන සරල කිරීම සඳහා K-Maps භාවිතා වේ.
- 2) K-Maps are graphical representations of truth tables.
K-Maps යනු සත්‍යතා වගු වල චිත්‍රණ නිරූපණයයි.
- 3) K-Maps help identify prime implicants and essential prime implicants.
K-Maps මූලික යෝග්‍යතා සහ අත්‍යවශ්‍ය මූලික යෝග්‍යතා (implicants) හඳුනා ගැනීමට උපකාරී වේ.
- 4) K-Maps are limited to simplifying expressions with up to four variables.
K-Maps විචල්‍ය හතරක් දක්වා ප්‍රකාශන සරල කිරීමට සීමා වේ.
- 5) K-Maps use adjacent cells to identify groups of terms for simplification.
K-Maps සරල කිරීම සඳහා පද කාණ්ඩ හඳුනා ගැනීමට යාබද කෝෂ භාවිතා කරයි.

Answer:4)

The statement saying K-Maps are limited to simplifying expressions with up to four variables is not true. K-Maps are most commonly used for expressions with up to four variables because they're easy to work with. However, they can technically handle more variables. The thing is, as you add more variables, using K-Maps becomes harder and less practical. So, while they're not strictly restricted to four variables, they work best for expressions with fewer variables.

කානෝ සිතියම් මගින් විවලය 4ක් දක්වා ප්‍රකාශන සුළු කිරීමට සීමාවේ යන ප්‍රකාශය අසත්‍යවේ. ගණනය කිරීමේ පහසුව නිසා කානෝ සිතියම් බහුලව භාවිතා වන්නේ විවලය 4ක් දක්වා ප්‍රකාශන සඳහා ය. කෙසේ වෙතත් ඒවාට තාක්ෂණික වශයෙන් වැඩි විවලය ප්‍රමාණයක් හැසිරවිය හැක. නමුත් කානෝ සිතියම් භාවිතයෙන් විවලයයන් ප්‍රමාණය වැඩි කරන විට එය සංකීර්ණ වී ප්‍රායෝගික නොවේ. එබැවින් ඒවා විවලය 4කට සීමා කර නොමැති අතර විවලයයන් අඩු ප්‍රකාශන සඳහා වඩාත් සුදුසු වේ.

6. What will be the output of the following code?
පහත කේතයේ ප්‍රතිදානය කුමක් වේවිද?

```
for i in range(2, 10, 2):
    for j in range(i):
        if j % 2 == 0:
            print(i, j, end=" ")
```

- 1) 2 0 4 0 4 2 6 0 6 2 8 0 8 2
- 2) 2 0 4 0 4 6 0 6 2 8 0 8 2
- 3) 2 0 4 0 4 2 6 0 8 0 8 2
- 4) 2 0 4 0 6 0 8 0
- 5) None of the above.

Answer : 5)

Here's the explanation. / පැහැදිලි කිරීම මෙන්න.

i Value	j Values Iterated	Output
2	0	20
4	0, 1, 2, 3	40
-	-	42
6	0, 1, 2, 3, 4, 5	60
-	-	62
-	-	64
8	0, 1, 2, 3, 4, 5, 6, 7	80
-	-	82
-	-	84
-	-	86

The output of the code is 2 0 4 0 4 2 6 0 6 2 6 4 8 0 8 2 8 4 8 6 because of how the nested loops and conditions work. For each value of i from the outer loop (2, 4, 6, 8), the inner loop iterates over values of j from 0 up to i-1. Only even values of j are printed. For i = 2, only 0 is even, resulting in 2 0. For i = 4, 0 and 2 are even, resulting in 4 0 and 4 2. For i = 6, 0, 2, and 4 are even, so it prints 6 0, 6 2 and 6 4. Finally, for i = 8, 0, 2, 4 and 6 are even, leading to 8 0, 8 2, 8 4 and 8 6.

nested ලූප සහ conditions ක්‍රියා කරන ආකාරය නිසා කේතයේ ප්‍රතිදානය 2 0 4 0 4 2 6 0 6 2 6 4 8 0 8 2 8 4 8 6 වේ. outer ලූපයෙන් (2, 4, 6, 8) i හි එක් එක් අගය සඳහා, අභ්‍යන්තර ලූපය 0 සිට i-1 දක්වා j අගයන් මත පුනරාවර්තනය වේ. මුද්‍රණය කර ඇත්තේ j හි ඉරට්ටේ අගයන් පමණි. i = 2 සඳහා, 0 පමණක් ඉරට්ටේ වේ. එහි ප්‍රතිඵලය වන්නේ 2 0 i = 4 සඳහා, 0 සහ 2 ඉරට්ටේ වේ, එහි ප්‍රතිඵලය වන්නේ 4 0 සහ 4 2 i = 6 සඳහා, 0 2 සහ 4 ඉරට්ටේ වේ, එබැවින් එය 6 0, 6 2 සහ 6 4 මුද්‍රණය කරයි. අවසාන වශයෙන්, i = 8 සඳහා, 0, 2, 4 සහ 6 ඉරට්ටේ වේ, එය 8 0, 8 2, 8 4 සහ 8 6 වෙත යොමු කරයි.

Therefore, The correct answer is, 5).
එබැවින්, නිවැරදි පිළිතුර නම්, 5) වේ.

7. What is the main advantage of using functions in programming?

ක්‍රමලේඛනයේදී ශ්‍රිත භාවිතා කිරීමේ ප්‍රධාන වාසිය කුමක්ද?

- 1) They increase the number of lines of code.
ඒවා කේත පේළි ගණන වැඩි කරයි.
- 2) They eliminate the need for control structures.
ඒවා පාලන ව්‍යුහයන් සඳහා ඇති අවශ්‍යතාවය ඉවත් කරයි.
- 3) They can only be defined in global scope.
ඒවා නිර්වචනය කළ හැක්කේ ගෝලීය විෂය පථය තුළ පමණි.
- 4) They help avoid code repetition and improve readability.
ඒවා කේත පුනරාවර්තනය වැළැක්වීමට සහ කියවීමේ හැකියාව වැඩි දියුණු කිරීමට උපකාරී වේ.
- 5) They always require return values.
ඒවාට සැමවිටම return අගයන් අවශ්‍ය වේ.

Answer : 4)

Functions in programming are self-contained blocks of code designed to perform a specific task. They help organize code into reusable pieces, making it more readable and maintainable.

ක්‍රමලේඛනයෙහි ශ්‍රිත යනු නිශ්චිත කාර්යයක් ඉටු කිරීම සඳහා නිර්මාණය කර ඇති self-contained කේත කොටස් වේ. ඒවා නැවත භාවිතා කළ හැකි කොටස් වලට කේතය සංවිධානය කිරීමට උපකාරී කරයි, එය වඩාත් කියවිය හැකි කර නඩත්තුව පහසු කරයි.

Let's go through the given options.

ලබා දී ඇති විකල්ප විමසා බලමු.

• **Option / විකල්පය 1)**

Incorrect. Functions typically reduce code repetition, which can lead to fewer lines of code.

වැරදි වේ. ශ්‍රිත සාමාන්‍යයෙන් කේත පුනරාවර්තනය අඩු කරයි, එමඟින් කේත රේඛා ගණන අඩු වීමට හේතු විය හැක.

• **Option / විකල්පය 2)**

Incorrect. Control structures (like loops and conditionals) are still needed inside or outside functions.

වැරදි වේ. පාලන ව්‍යුහයන් (පුනරාවර්තන සහ කොන්දේසි වැනි) තවමත් ශ්‍රිත ඇතුළත හෝ පිටත ක්‍රියාකාරකම් සඳහා අවශ්‍ය වේ.

• **Option / විකල්පය 3)**

Incorrect. Functions can be defined in both global and local scopes depending on the programming language.

වැරදි වේ. ක්‍රමලේඛන භාෂාව මත පදනම්ව global සහ local විෂය පථයන් දෙකෙහිම ශ්‍රිත අර්ථ දැක්විය හැක.

• **Option / විකල්පය 4)**

This is the main advantage. Functions allow you to reuse code without repeating it, making the program easier to read and maintain.

මෙය ප්‍රධාන වාසියයි. ශ්‍රිත මඟින් ඔබට කේතය නැවත නැවත නොලියා භාවිතා කිරීමට ඉඩ සලසයි, එමඟින් වැඩසටහන කියවීම සහ නඩත්තු කිරීම පහසු වේ.

• **Option / විකල්පය 5)**

Incorrect. Functions can return values, but it's not always necessary for them to return something; some functions just perform tasks without returning anything.

වැරදි වේ. ශ්‍රිතවලට අගයන් return කළ හැක, නමුත් යමක් return කිරීමට සැමවිටම අවශ්‍ය නොවේ සමහර ශ්‍රිත කිසිවක් return ලෙස ලබා නොදේ.

Therefore, the answer is, 4).

එබැවින් පිළිතුර 4) වේ.

8. What will be the output of the following Python code?
පහත පයිතන් කේතයේ ප්‍රතිඵලය කුමක් වේවිද?

```
letters = [1, 2, 3, 4, 5]
result = 1
for num in letters:
    result *= num
    if num % 2 == 0:
        result += 1
print(result)
```

- 1) 15 2) 30 3) 180 4) 185 5) 100

Answer : 4)

Explanation / පැහැදිලි කිරීම

This Python code initializes a list `letters = [1,2,3,4,5]` and a variable `result` with a value of 1. It then iterates over each element in the list using a for loop. For each number in the list, `result` is multiplied by the current number (`num`).

මෙම පයිතන් කේතය මඟින් ලැයිස්තු `letters = [1,2,3,4,5]` සහ 1 අගයක් සහිත විචල්‍ය ප්‍රතිඵලයක් ආරම්භ කරයි. ඉන්පසු එය for loop එකක් භාවිතයෙන් ලැයිස්තුවේ ඇති එක් එක් මූලද්‍රව්‍ය මත පුනරාවර්තනය වේ. ලැයිස්තුවේ සෑම අංකයක් සඳහාම, ප්‍රතිඵලය වත්මන් අංකයෙන් (`num`) ගුණ කරනු ලැබේ.

Additionally, if the current number is even (determined by the condition `num % 2 == 0`), 1 is added to `result`. අතිරේකව, වත්මන් අංකය ඉරට්ටේ නම් (කොන්දේසි `num % 2 == 0`), 1 ප්‍රතිඵලය වෙත එකතු වේ.

The process unfolds as follows, in the first iteration with `num = 1`, `result` remains 1. In the second iteration with `num = 2`, `result` is multiplied by 2 to give 2, then incremented by 1 due to the number being even, resulting in 3. In the third iteration, `num = 3`, so `result` is multiplied by 3, giving 9, with no increment since 3 is odd. In the fourth iteration, `num = 4`, `result` becomes 36 and is incremented by 1 (since 4 is even), resulting in 37.

ක්‍රියාවලිය පහත පරිදි දිග හැරේ, පළමු පුනරාවර්තනයේ `num = 1` සමඟින්, ප්‍රතිඵලය 1 ලෙස පවතී. දෙවන පුනරාවර්තනයේ `num = 2` සමඟ, ප්‍රතිඵලය 2 න් ගුණ කර 2 ලබා දෙයි, ඉන්පසු සංඛ්‍යාව ඉරට්ටේ වීම නිසා 1 කින් වැඩි වේ 3. තුන්වන පුනරාවර්තනයේ, `num = 3`, එබැවින් ප්‍රතිඵලය 3 න් ගුණ කරනු ලැබේ, 9 ලබා දෙයි, 3 ඔත්තේ බැවින් වැඩි වීමක් නොමැත. සිව්වන පුනරාවර්තනයේ, `num = 4`, ප්‍රතිඵලය 36 බවට පත් වන අතර 1 කින් වැඩි වේ (4 ඉරට්ටේ බැවින්), ප්‍රතිඵලය 37 වේ.

Finally, in the fifth iteration, `num = 5` leads to `result` being multiplied by 5, resulting in 185. Since 5 is odd, no increment occurs, and the loop concludes with the final value of `result` being 185. Therefore, the output of the program is 185.

අවසාන වශයෙන්, පස්වන පුනරාවර්තනයේ දී, `num = 5` ප්‍රතිඵලය 5 න් ගුණ කිරීමට හේතු වේ, ප්‍රතිඵලය 185. 5 ඔත්තේ බැවින්, වර්ධනයක් සිදු නොවේ, සහ ප්‍රතිඵලයේ අවසාන අගය 185 සමඟ ලූපය අවසන් වේ. එබැවින්, ප්‍රතිදානය 185 වේ.

Therefore, the correct answer is, 4).

එබැවින්, නිවැරදි පිළිතුර වන්නේ, 4) වේ.

9. Which of the following statements about Python lists is correct?

පයිතන් ලැයිස්තු පිළිබඳ නිවැරදි ප්‍රකාශය කුමක්ද?

- 1) A Python list can only contain elements of the same data type.
පයිතන් ලැයිස්තුවක ඇතුළත් කළ හැක්කේ එකම දත්ත වර්ගයක පමණි.
- 2) The size of a Python list must be specified at the time of its creation.
පයිතන් ලැයිස්තුවක් නිර්මාණය කරන විට එහි ප්‍රමාණය නියම කළ යුතුය.
- 3) Python lists are immutable, meaning their elements cannot be changed.
පයිතන් ලැයිස්තු අවිධියෙන් පවතින අතර එහි අංග වෙනස් කළ නොහැක.
- 4) A Python list can contain elements of different data types.
පයිතන් ලැයිස්තුවක් විවිධ දත්ත වර්ග අඩංගු කළ හැක.
- 5) Once a Python list is created, you cannot add or remove elements from it.
පයිතන් ලැයිස්තුවක් නිර්මාණය වූ විට එහි අංග එකතු හෝ ඉවත් කළ නොහැක.

Answer : 4)

Let's review each statement to determine which one is correct.

කුමන ප්‍රකාශය නිවැරදි දැයි තීරණය කිරීමට අපි එක් එක් ප්‍රකාශය සමාලෝචනය කරමු.

Statement / ප්‍රකාශන 1)

This is false statement. A Python list can contain elements of different data types. For example, you can have a list with integers, strings, and even other lists.

මෙය අසත්‍ය ප්‍රකාශනයක් වේ. පයිතන් ලැයිස්තුවක විවිධ දත්ත වර්ගවල මූලාංග අඩංගු විය හැක. උදාහරණයක් ලෙස, ඔබට පූර්ණ සංඛ්‍යා, string සහ වෙනත් ලැයිස්තු සහිත ලැයිස්තුවක් තිබිය හැක.

Statement / ප්‍රකාශන 2)

This is false statement. Python lists are dynamic in size, meaning you do not need to specify the size when creating a list. You can add or remove elements as needed.

මෙය අසත්‍ය ප්‍රකාශනයක් වේ. පයිතන් ලැයිස්තු ගතික ප්‍රමාණයෙන් යුක්ත වේ, එනම් ලැයිස්තුවක් සෑදීමේදී ඔබට ප්‍රමාණය සඳහන් කිරීමට අවශ්‍ය නොවේ. ඔබට අවශ්‍ය පරිදි මූලාංග එකතු කිරීමට හෝ ඉවත් කිරීමට හැකිය.

Statement / ප්‍රකාශන 3)

This is false statement. Python lists are mutable, meaning their elements can be changed after the list is created.

මෙය අසත්‍ය ප්‍රකාශනයක් වේ. පයිතන් ලැයිස්තු mutable වේ, එනම් ලැයිස්තුව නිර්මාණය කිරීමෙන් පසු ඒවායේ මූලාංග වෙනස් කළ හැකිය.

Statement / ප්‍රකාශන 4)

This is true statement. A Python list can store elements of any data type, such as integers, strings, floats, and even other lists.

මෙය සත්‍ය ප්‍රකාශනයක් වේ. පයිතන් ලැයිස්තුවකට පූර්ණ සංඛ්‍යා, string, float සහ වෙනත් ලැයිස්තු වැනි ඕනෑම දත්ත වර්ගයක මූලාංග ගබඩා කළ හැක.

Statement / ප්‍රකාශන 5)

This is false statement. Python lists allow you to add, remove, or modify elements after the list has been created.

මෙය අසත්‍ය ප්‍රකාශනයක් වේ. පයිතන් ලැයිස්තු මගින් ඔබට ලැයිස්තුව නිර්මාණය කිරීමෙන් පසු මූලාංග එක් කිරීමට, ඉවත් කිරීමට හෝ වෙනස් කිරීමට ඉඩ ලබා දේ.

So, the correct answer is 4).

එබැවින්, නිවැරදි පිළිතුර 4).

10. What will be the output of the following Python code?

පහත පයිතන් කේතයේ ප්‍රතිදානය කුමක් වේවිද?

```
def my_func(x=[]):
    x.append(len(x))
    return x
```

```
print(my_func())
print(my_func())
```

- 1) [0] 2) [0] 3) [0] 4) [1] 5) [1]
 [0, 0] [0, 1] [1, 1] [1, 1] [1, 0]

Answer : 2)

Explanation. / පැහැදිලි කිරීම

In the function `my_func`, the default argument `x` is a mutable list.

`my_func` ශ්‍රිතයේ, පෙරනිමි තර්කය `x` යනු විකෘති ලැයිස්තුවකි.

When the function is called for the first time with `my_func()`, it initializes `x` as an empty list and appends the length of `x`, which is 0, resulting in `[0]`.

`my_func()` සමඟ ශ්‍රිතය පළමු වරට කැඳවූ විට, එය `x` හිස් ලැයිස්තුවක් ලෙස ආරම්භ කර `x` හි දිග එකතු කරයි, එය 0 වන අතර, එහි ප්‍රතිඵලයක් ලෙස `[0]`.

On the second call to `my_func()`, it uses the same list from the first call, now containing `[0]`.

`my_func()` වෙත දෙවන ඇමතුමේදී, එය පළමු ඇමතුමෙන් එකම ලැයිස්තුව භාවිතා කරයි, දැන් `[0]` අඩංගු වේ.

The length of `x` is 1, so it appends 1, resulting in `[0,1]`.

`x` හි දිග 1 වේ, එබැවින් එය 1 අනුයුක්ත කරයි, ප්‍රතිඵලයක් ලෙස `[0,1]`.

Therefore, the first call outputs `[0]`, and the second call outputs `[0,1]`.

එමනිසා, පළමු ඇමතුම ප්‍රතිදානය කරයි `[0]`, සහ දෙවන ඇමතුම ප්‍රතිදානය `[0,1]`.

Therefore, the correct answer is, 2).

එබැවින්, නිවැරදි පිළිතුර වන්නේ, 2) වේ.