# Operators .

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Arithmetic operators | | | Comparison operators | | | Boolean operators | | |
| Python code | Meaning | Pseudocode | Python code | Meaning | Pseudocode | Python code | Meaning | Pseudocode |
| \* | Multiplication | \* | == | Equal to | is equal to | and | And | AND |
| / | Division (real) | / | != | Not equal to | is not equal to | or | Or | OR |
| + | Addition | + | > | Greater than | is greater than | not | Not | NOT |
| - | Subtraction | - | < | Less than | is less than |  |  |  |
| % | Modulo | MOD | <= | Less than or equal to | is less than or equal to |  |  |  |
| \*\* | To the power of | \*\* | >= | Greater than or equal to | is greater than or equal to |  |  |  |
| // | Integer division | DIV |  |  |  |  |  |  |

# Programming concepts or constructs .

|  |  |  |  |
| --- | --- | --- | --- |
| Concept / construct | Python code | Pseudocode | Example |
| Commenting |  |  |  |
| Comment on 1 line | # | # | # multiples age by year  answer <- age \* year |
| Comment over several lines | """,""" | """,""" | """  I need a long comment  That goes over several lines  """ |
| Variables |  |  |  |
| Assignment | = | <- | answer <- age \* year |
| Constant | MY\_CONSTANT | MY\_CONSTANT | BIRTH\_YEAR <- 2005 |
| Variables | my\_variable | my\_variable | not\_guessed <- True |
| Input/output |  |  |  |
| User input | input() | USERINPUT | age <- USERINPUT |
| Output | print() | OUTPUT | OUTPUT age  OUTPUT “string” |
| Casting |  |  |  |
| Converting to another data type | str()  int()  float() | STRING()  INTEGER()  FLOAT() | STRING(age)  INTEGER(“5”)  INTEGER(age)  FLOAT(“5.2”)  FLOAT(average) |
| Selection |  |  |  |
| if | if answer == correct:  print("Correct!") | IF condition THEN  block of statements  ENDIF | IF answer is equal to guess THEN  OUTPUT “Correct!”  ENDIF |
| if, else | if answer == correct:  print("Correct!")  else:  print("Incorrect!") | IF condition THEN  block of statements  ELSE  block of statements  ENDIF | IF answer is equal to guess THEN  OUTPUT “Correct!”  ELSE  OUTPUT “Incorrect!”  ENDIF |
| if, elif, else | if answer == 5:  print("Correct!")  elif answer == 6:  print("Almost")  else:  print("Incorrect!") | IF condition THEN  block of statements  ELSEIF condition THEN  block of statements  ELSE  block of statements  ENDIF | IF answer is equal to 5 THEN  OUTPUT “Correct!”  ELSEIF answer is equal to 6 THEN  OUTPUT “Almost”  ELSE  OUTPUT “Incorrect!”  ENDIF |
| Iteration |  |  |  |
| For loop | for i in range(5):  print(i)  for letter in word:  print(letter) | FOR … in … to …  block of statements  ENDFOR | FOR i = 1 to 10  OUTPUT i  ENDFOR  FOR item in list  OUTPUT item  ENDFOR  FOR letter in word  OUTPUT letter  ENDFOR |
| While loop | while not\_guessed:  number = input() | WHILE condition  block of statements  ENDWHILE | WHILE not\_guessed  number <- USERINPUT  ENDWHILE |
| Subroutines |  |  |  |
| Define a subroutine | def add(a, b):  answer = a + b  print (answer) | SUBROUTINE identifier(parameters)  block of statements  ENDSUBROUTINE | SUBROUTINE add(1, b)  answer <- a + b  OUTPUT answer  ENDSUBROUTINE |
| Use return in a function | return answer | RETURN value | RETURN answer |
| Call a subroutine | add(3, 4) | identifier(parameters) | add(3, 4) |
| Lists |  |  |  |
| Assignment | animals = ["Dog","Cow"] | identifier <- [item1, item2] | animals <- ["Dog","Cow"] |
| Accessing an item | animals[0] | identifier[index] | animals[0] |
| Updating an item | animals[0] = "Sheep" | identifier[index] <- item | animals[0] <- “Sheep” |
| Length of list | len(animals) | LEN(identifier) | LEN(animals) |
| String handling |  |  |  |
| Length of string | len(word) | LEN(identifier) | LEN(word) |
| Index of a character | word = "HELLO"  word.index("E") | identifier.INDEX(char) | word.INDEX(“L”) |
| Get the character at a location | word[0] | identifier[index] | word[0] |
| Substring | word[1:3]  word[1:]  word[:3] | identifier[index:index]  Identifier[index:]  identifier[:index] | word[1:3]  word[1:]  word[:3] |