***第3章学習11\_展開1\_ワークシート***

**論理演算の仕組みと図と表で表現してみよう**

　　年　　組(　　)　　　　　　　　　.

**（１）コンピュータの演算は四則演算とは異なり，ＡＮＤ，ＯＲ，ＮＯＴの３つの論理演算の組み合わせで行われている。０を「なし」，１を「あり」と考え，ＡＮＤ，ＯＲ，ＮＯＴをベン図と表で表現して，その具体例を考えてみよう。**

|  |  |  |  |
| --- | --- | --- | --- |
|  | ベン図 | 表 | 具体例 |
| Ａ　and B  （ＡＮＤ） |  | |  |  |  | | --- | --- | --- | | A | B |  | | 0 | 0 |  | | 0 | 1 |  | | 1 | 0 |  | | 1 | 1 |  | |  |
| Ａ　or B  （ＯＲ） |  | |  |  |  | | --- | --- | --- | | A | B |  | | 0 | 0 |  | | 0 | 1 |  | | 1 | 0 |  | | 1 | 1 |  | |  |
| Not A  （ＮＯＴ） |  | |  |  | | --- | --- | | A |  | | 0 |  | | 1 |  | |  |

**（２）上の表のベン図と表におけるＡＮＤとNOTを組み合わせたＮＡＮＤの場合，どのようなベン図と表になるか表現してみよう。**

|  |  |  |
| --- | --- | --- |
|  | ベン図 | 表 |
| ＮＡＮＤ |  | |  |  |  | | --- | --- | --- | | A | B |  | | 0 | 0 |  | | 0 | 1 |  | | 1 | 0 |  | | 1 | 1 |  | |

**論理演算の仕組みと図と表で表現してみよう**

教師用

　　年　　組(　　)　　　　　　　　　.

【指導ポイント】　具体例を挙げて説明する。

※Ａを高校の先生である，Ｂを男性の先生であるとした場合，Ａ and Bは高校の男性の先生であるといったように具体例を挙げて説明する。OR，NOTは自分なりに具体例を考えせる

**（１）コンピュータの演算は四則演算とは異なり，ＡＮＤ，ＯＲ，ＮＯＴの３つの論理演算の組み合わせで行われている。０を「なし」，１を「あり」と考え，ＡＮＤ，ＯＲ，ＮＯＴをベン図と表で表現して，その具体例を考えてみよう。**

|  |  |  |  |
| --- | --- | --- | --- |
|  | ベン図 | 表 | 具体例 |
| Ａ　and B  （ＡＮＤ） |  | |  |  |  | | --- | --- | --- | | A | B |  | | 0 | 0 | 0 | | 0 | 1 | 0 | | 1 | 0 | 0 | | 1 | 1 | 1 | | Ａを高校の先生  Ｂを男性の先生  ならＡ and Bは  高校の男性の先生  となる。 |
| Ａ　or B  （ＯＲ） |  | |  |  |  | | --- | --- | --- | | A | B |  | | 0 | 0 | 0 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 1 | | Ａを高校の先生  Ｂを男性の先生  ならＡ or Bは  高校の先生か男性の先生となる。 |
| Not A  （ＮＯＴ） |  | |  |  | | --- | --- | | A |  | | 0 | 1 | | 1 | 0 | | Ａを高校の先生  ならNot Aは  高校の先生ではないとなる。 |

【評価ポイント】

・自分なりに考えた具体例に置き換えて表現できているか。

・十分な量が書かれているか。

**（２）上の表のベン図と表におけるＡＮＤとNOTを組み合わせたＮＡＮＤの場合，どのようなベン図と表になるか表現してみよう。**

|  |  |  |
| --- | --- | --- |
|  | ベン図 | 表 |
| ＮＡＮＤ |  | |  |  |  | | --- | --- | --- | | A | B |  | | 0 | 0 | 1 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 0 |   【指導ポイント】  活動につまずいている生徒に，Small Stepで指導する。（まずはＡ and Bのベン図を書かせ，その後に反転させるように，１つ１つスモールステップで考えさせるように支援する。） |

***第3章学習11\_展開2\_ワークシート***

**プログラミング言語を用いて**

**２つの数の計算誤差の問題を解決しよう**

　　年　　組(　　)　　　　　　　　　.

**（１）整数同士の手計算の結果とプログラミングによる計算結果を比較し，どのような時に計算誤差が発生するか，調べてみよう。**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 計算式 | 手計算の  結果（Ｘ） | プログラミングによる  計算結果（Ｙ） | 計算誤差 |
| 整数の  加算 | 55+30 |  |  |  |
| 整数の  減算 |  |  |  |  |
| 小数点の  加算 |  |  |  |  |
| 小数点の  減算 |  |  |  |  |

**（２）どのような時に計算誤差が発生しているか，その条件を見つけ，誤差が起こる理由について考えてみよう。**

|  |  |
| --- | --- |
| 計算誤差が発生する条件 | 計算誤差が起こる理由 |
|  |  |

**（３）計算誤差を無くすためにどのようにすればいいか解決方法を考え，プログラミングで実行してみよう。**

|  |  |
| --- | --- |
| 計算誤差を解決する方法 | 作成したプログラミング |
|  |  |

**プログラミング言語を用いて**

教師用

**２つの数の計算誤差の問題を解決しよう**

　　年　　組(　　)　　　　　　　　　.

【指導ポイント】

誤差が発生する条件を発見させる。

※いきなり計算誤差が発生する計算式を与えるのではなく，自分たちで計算を数回させ，どのような条件で計算誤差が発生するのか発見させる。

**（１）整数同士の手計算の結果とプログラミングによる計算結果を比較し，どのような時に計算誤差が発生するか，調べてみよう。**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 計算式 | 手計算の  結果（Ｘ） | プログラミングによる  計算結果（Ｙ） | 計算誤差 |
| 整数の  加算 | 55+30 | 85 | 85 | 誤差なし |
| 整数の  減算 | 24-20 | ４ | ４ | 誤差なし |
| 小数点の  加算 | 1.05+1.02 | 2.07 | 2.0700000000000003 | 誤差あり |
| 小数点の  減算 | 0.28-0.27 | 0.01 | 0.010000000000000009 | 誤差あり |

【指導ポイント】

理由が出てこない場合は，オーバーフローの起こるプログラムを実行してみる。

※計算誤差の仕組みの理由を見つけるのは困難なため，オーバーフローが起こるプログラムを実行して示すなどして，支援する。

**（２）どのような時に計算誤差が発生しているか，その条件を見つけ，誤差が起こる理由について考えてみよう。**

|  |  |
| --- | --- |
| 計算誤差が発生する条件 | 計算誤差が起こる理由 |
| 小数点を含んだ計算を行った時に  小数点以下の部分に計算誤差が発生する。 | 小数点以下の数値の扱える桁数以上の数字を切り捨てて計算するため，誤差が発生する。 |

**（３）計算誤差を無くすためにどのようにすればいいか解決方法を考え，プログラミングで実行してみよう。**

|  |  |
| --- | --- |
| 計算誤差を解決する方法 | 作成したプログラミング |
| * 変数の型を変えて計算を行う。 * 一度整数にして計算し，計算した後小数にする | x = 0.28-0.27  x =round(x\*100)  x = x/100  print(x) |

【評価ポイント】

計算誤差を解決する方法を複数発見することができたか。

***第3章学習12\_展開1,2,3\_ワークシート***

**センサやアクチュエータを使って問題点を解決してみよう**

　　年　　組(　　)　　　　　　　　　.

**（１）センサとアクチュエータを使って身近な問題点を解決する方法を考えてみよう。**

|  |  |
| --- | --- |
| 身近な問題点 |  |
| 問題点の解決方法 |  |
| センサやアクチュエータを  どのように活用するか |  |

**（２）プログラムに発生した問題点をどのように解決すればいいか考えてみよう。**

micro: bitで右と左と真ん中の傾きを検出するプログラムを作成しようとした。加速度センサが右に傾いているデータを検出すればＬＥＤに「R」が点灯し，そうでなければLEDに「-」が点灯するプログラムは作成できたが，左の傾きを検出するプログラムができていない。左の傾きを検出したらＬＥＤに「Ｌ」を表示させるプログラムを作成しなさい。

|  |  |
| --- | --- |
|  |  |

「はじめようmicro:bit」（https://microbit.org/ja/guide/）を加工して作成

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | from microbit import \* #マイクロビット用モジュールの読み込み  while True: #ずっと繰り返す  reading = accelerometer.get\_x() #加速度センサのx軸の傾きを読む  if reading > 100: # X軸の傾きが100より大きいなら  display.show("R") #ＬＥＤに「Ｒ」を表示  else: #そうでなければ  display.show("-") #ＬＥＤに「－」を表示 |

**（３）センサから入力された値に応じて動きが変わるプログラムを制作しよう。**

micro: bitで右と左と真ん中の傾きを検出するプログラムを作成したが，ｘ軸の傾きが150の時も

，ｘ軸の傾きが300の時も，ＬＥＤには同じ「Ｒ」が表示されるだけである。x軸の傾きの度合いが150と300ではＬＥＤの表示結果が異なるプログラムを制作しなさい。

**センサやアクチュエータを使って問題点を解決してみよう**

教師用

　　年　　組(　　)　　　　　　　　　.

**（１）センサとアクチュエータを使って身近な問題点を解決する方法を考えてみよう**

|  |  |
| --- | --- |
| 身近な問題点 | 鍵を貸し出し時間以外に借りられるため，管理できていない。 |
| 問題点の解決方法 | 貸し出し時間以外に鍵を借りられたらアラートを鳴らす。 |
| センサやアクチュエータを  どのように活用するか | 超音波センサで鍵を検知し，貸し出し時間以外になくなったらアラートを鳴らす。 |

【指導ポイント】

身近な問題から解決する方法を考えさせる。

※ブレーンストーミングやマンダラートなどを活用し，身近な問題点を発想させてから，問題点の解決方法を考えさせるようにする。

**（２）プログラムに発生した問題点をどのように解決すればいいか考えてみよう**

micro: bitで右と左と真ん中の傾きを検出するプログラムを作成しようとした。加速度センサが右に傾いているデータを検出すればＬＥＤに「R」が点灯し，そうでなければLEDに「-」が点灯するプログラムは作成できたが，左の傾きを検出するプログラムができていない。左の傾きを検出したらＬＥＤに「Ｌ」を表示させるプログラムを作成しなさい。

|  |  |
| --- | --- |
|  |  |

「はじめようmicro:bit」（https://microbit.org/ja/guide/）を加工して作成

【指導ポイント】

フローチャートを描いて，条件分岐の構造を可視化する。

※いきなりプログラミングを行うのではなく，まずはフローチャートでどのように条件分岐するのか考えさせる。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | from microbit import \* #マイクロビット用モジュールの読み込み  while True: #ずっと繰り返す  reading = accelerometer.get\_x() #加速度センサのx軸の傾きを読む  if reading > 100: # X軸の傾きが100より大きいなら  display.show("R") #ＬＥＤに「Ｒ」を表示  elif reading < -100: 　# X軸の傾きが-100より小さいなら  display.show("L") #ＬＥＤに「L」を表示  else: #そうでなければ  display.show("-") #ＬＥＤに「－」を表示 |

**（３）センサから入力された値に応じて動きが変わるプログラムを制作しよう。**

micro: bitで右と左と真ん中の傾きを検出するプログラムを作成したが，ｘ軸の傾きが150の傾きも，ｘ軸の傾きが300の時も，ＬＥＤには同じ「Ｒ」が表示されるだけである。x軸の傾きの度合いが150と300ではＬＥＤの表示結果が異なるプログラムを制作しなさい。

【評価ポイント】

センサの値の変化に応じてLEDが適切に表示されるようにプログラムすることができたか。

***第3章学習13\_展開1\_ワークシート***

**条件によって処理が変化するプログラムを作成しよう**

　　年　　組(　　)　　　　　　　　　.

**プログラムでは条件によって処理が変化するものもある。これを分岐と呼ぶ。具体的な例としては「テストの点数が60点以上なら合格，60点未満なら不合格と表示する」プログラムなどが挙げられる。**

**（１）分岐となる処理にはどのようなものがあるか考えよう。**

|  |  |  |
| --- | --- | --- |
| 条件 | Yes | No |
| (例)  テストの点数が60点以上 | 合格と表示する | 不合格と表示する |
|  |  |  |

**（２）点数xが60以上なら合格と表示する次のプログラムを作成し，点数xを変更することで表示が変化することを確認しよう。**

|  |  |
| --- | --- |
| 1  2  3  4  5 | x = 80 #x=80とする  if x >= 60: 　 #xが60以上のときの処理  print("合格")  else: #xが60以上でないときの処理  print("不合格") |

|  |  |
| --- | --- |
| 点数xの値 | どのように表示されるか |
| 例）　60 | 合格と表示される。 |
|  |  |

**（３）先ほどのプログラムを改良し，点数xが30未満なら再試験，そうでないならば合格と表示するプログラムを作成しよう。また正しく処理を変更できているか確認しよう。**

|  |  |
| --- | --- |
| 点数xの値 | どのように表示されるか |
|  |  |
|  |  |

**条件によって処理が変化するプログラムを作成しよう**

教師用

　　年　　組(　　)　　　　　　　　　.

**プログラムでは条件によって処理が変化するものもある。これを分岐と呼ぶ。具体的な例としては「テストの点数が60点以上なら合格，60点未満なら不合格と表示する」プログラムなどが挙げられる。**

**（１）分岐となる処理にはどのようなものがあるか考えよう。**

|  |  |  |
| --- | --- | --- |
| 条件 | Yes  【指導ポイント】  数値の大小で処理が分かれるような例を考えさせる。  ※他の生徒と相談して考えるよう促す。 | No |
| (例)  テストの点数が60点以上 | 合格と表示する | 不合格と表示する |
| 賞味期限が過ぎていないか | 商品を廃棄する | 商品を商品棚に置いておく |

**（２）点数xが60以上なら合格と表示する次のプログラムを作成し，点数xを変更することで表示が変化することを確認しよう。**

|  |  |
| --- | --- |
| 1  2  3  4  5 | x = 80 #x=80とする  if x >= 60: 　 #xが60以上のときの処理  print("合格")  else: #xが60以上でないときの処理  print("不合格") |

|  |  |
| --- | --- |
| 点数xの値 | どのように表示されるか  【指導ポイント】  不合格と表示されるような数値を生徒が数値変更することで確認をさせる。 |
| 例）　60 | 合格と表示される。 |
| 30 | 不合格と表示される。 |

【指導ポイント】

処理が数値によって変化することを生徒自身に確認させる。

**（３）先ほどのプログラムを改良し，点数xが30未満なら再試験，そうでないならば合格と表示するプログラムを作成しよう。また正しく処理を変更できているか確認しよう。**

|  |  |
| --- | --- |
| 点数xの値 | どのように表示されるか |
| 29 | 再試験と表示される。 |
| 30 | 合格と表示される。 |

**x = 40**

【評価ポイント】

点数の値と表示のされ方が正しく対応しているか。

**if x < 30 :**

**print("再試験")**

**else :**

**print("合格")**

***第3章学習13\_展開2\_ワークシート***

**同じ処理を繰り返すプログラムを作成しよう**

　　年　　組(　　)　　　　　　　　　.

**プログラムでは同じ処理を繰り返す構造を持つものもあり，これを反復と呼ぶ。具体的な例としては「ある値を10ずつ増やす処理を5回繰り返す」プログラムなどがあげられる。**

**（１）「ある値を10ずつ増やす処理を5回繰り返す」プログラムの処理の流れをまとめよう。**

|  |  |
| --- | --- |
| 繰り返す回数 | 合計の値 x |
| 繰り返す前 | 0 |
| 1 | 10 |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |

**（２）「ある値を10ずつ増やす処理を5回繰り返す」プログラムを作成し，上でまとめた結果と一致することを確認しよう。**

|  |  |
| --- | --- |
| 1  2  3  4  5 | x = 0  print("x = " , x)  for i in range(1, 6, 1): #1から6未満の間1ずつカウントを増やしながら繰り返す  x = x + 10  print("x = " , x) |

**（３）先ほどのプログラムを改良し，「ある値を繰り返した回数iずつ増やす処理を5回繰り返す」プログラムを作成し，処理の流れをまとめよう。なお繰り返す前のxの値は0とする。**

|  |  |
| --- | --- |
| 繰り返す回数 | 合計の値x |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 | 15 |

**同じ処理を繰り返すプログラムを作成しよう**

教師用

　　年　　組(　　)　　　　　　　　　.

**プログラムでは同じ処理を繰り返す構造を持つものもあり，これを反復と呼ぶ。具体的な例としては「ある値を10ずつ増やす処理を5回繰り返す」プログラムなどがあげられる。**

**（１）「ある値を10ずつ増やす処理を5回繰り返す」プログラムの処理の流れをまとめよう。**

【指導ポイント】

生徒に演習させるだけではなく，1行ずつ何が行われているか説明する。

【指導ポイント】

繰り返すことで値が変化する様子は丁寧に説明するとよい。

|  |  |
| --- | --- |
| 繰り返す回数 | 合計の値 x |
| 繰り返す前 | 0 |
| 1 | 10 |
| 2 | 20 |
| 3 | 30 |
| 4 | 40 |
| 5 | 50 |

**（２）「ある値を10ずつ増やす処理を5回繰り返す」プログラムを作成し，上でまとめた結果と一致することを確認しよう。**

|  |  |
| --- | --- |
| 1  2  3  4  5 | x = 0  print("x = " , x)  for i in range(1, 6, 1): #1から6未満の間1ずつカウントを増やしながら繰り返す  x = x + 10  print("x = " , x) |

**（３）先ほどのプログラムを改良し，「ある値を繰り返した回数iずつ増やす処理を5回繰り返す」プログラムを作成し，処理の流れをまとめよう。なお繰り返す前のxの値は0とする。**

【評価ポイント】

繰り返す回数に応じて合計の値ｘが正しく記載されているか。

【指導ポイント】

5回が難しい場合は2,3回の繰り返しから作成をさせる。

|  |  |
| --- | --- |
| 繰り返す回数 | 合計の値x |
| 1 | 1 (0+1) |
| 2 | 3　(1+2) |
| 3 | 6 (3+3) |
| 4 | 10 (6+4) |
| 5 | 15 |

**x = 0**

**print("x = " , x)**

**for i in range(1, 6, 1):**

**x = x + i**

**print("x = " , x)**

***第3章学習13\_展開3\_ワークシート***

**分岐と反復を組み合わせて複雑なプログラムを作成しよう**

　　年　　組(　　)　　　　　　　　　.

**分岐と反復を組み合わせると，「繰り返していく中である条件を満たす場合だけ特定の処理を行う」ことが可能となり，アルゴリズムの表現の幅を広げることができるようになる。**

**（１）「ある値を10ずつ増やす処理を5回繰り返すが，繰り返す回数が奇数回の場合だけ画面に表示する」プログラムの処理の流れをまとめよう。**

|  |  |  |
| --- | --- | --- |
| 繰り返す回数 | 合計の値 x | 画面表示するか |
| 繰り返す前 | 0 | しない |
| 1 | 10 | する |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |

**（２）「ある値を10ずつ増やす処理を5回繰り返すが，繰り返す回数が奇数回の場合だけ画面に表示する」プログラムを作成し，上でまとめた結果と一致することを確認しよう。**

|  |  |
| --- | --- |
| 1  2  3  4  5 | x = 0  for i in range(1, 6, 1):  x = x + 10  if i%2==1: #iを2で割ったときの余りが1のとき  print("x = " , x) |

**（３）先ほどのプログラムを変更し，「繰り返す回数が偶数回の場合だけ画面に表示する」プログラムを作成し，処理の流れをまとめよう。なお繰り返す前のxの値は0とする。**

|  |  |  |
| --- | --- | --- |
| 繰り返す回数 | 合計の値x | 画面表示するか |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |

**分岐と反復を組み合わせて複雑なプログラムを作成しよう**

教師用

　　年　　組(　　)　　　　　　　　　.

**分岐と反復を組み合わせると，「繰り返していく中である条件を満たす場合だけ特定の処理を行う」ことが可能となり，アルゴリズムの表現の幅を広げることができるようになります。**

【指導ポイント】

繰り返すことで値が変化することを丁寧に説明する。

**（１）「ある値を10ずつ増やす処理を5回繰り返すが，繰り返す回数が奇数回の場合だけ画面に表示する」プログラムの処理の流れをまとめよう。**

|  |  |  |
| --- | --- | --- |
| 繰り返す回数 | 合計の値 x | 画面表示するか |
| 繰り返す前 | 0 | しない |
| 1 | 10 | する |
| 2 | 20 | しない |
| 3 | 30 | する |
| 4 | 40 | しない |
| 5 | 50 | する |

**（２）「ある値を10ずつ増やす処理を5回繰り返すが，繰り返す回数が奇数回の場合だけ画面に表示する」プログラムを作成し，上でまとめた結果と一致することを確認しよう。**

【指導ポイント】

4行目について強調して説明する。

|  |  |
| --- | --- |
| 1  2  3  4  5 | x = 0  for i in range(1, 6, 1):  x = x + 10  if i%2==1: #iを2で割ったときの余りが1のとき  print("x = " , x) |

【指導ポイント】

5回が難しい場合は2,3回の繰り返しから作成させる。

**（３）先ほどのプログラムを変更し，「繰り返す回数が偶数回の場合だけ画面に表示する」プログラムを作成し，処理の流れをまとめよう。なお繰り返す前のxの値は0とする。**

【評価ポイント】

合計の値ｘと画面表示するかの列が正しく記載されているか。

|  |  |  |
| --- | --- | --- |
| 繰り返す回数 | 合計の値x | 画面表示するか |
| 1 | 10 | しない |
| 2 | 20 | する |
| 3 | 30 | しない |
| 4 | 40 | する |
| 5 | 50 | しない |

**x = 0**

**for i in range(1, 6, 1):**

**x = x + 10**

**if i%2==0:**

**print("x = " , x)**

***第3章学習14\_展開1ワークシート***

**リストを使ってデータをまとめて処理をしよう**

　　年　　組(　　)　　　　　　　　　.

**プログラムが大規模になると扱うデータの数も膨大になる。ここではデータを一つのかたまりとして処理をするためのリストについて学ぼう。**

**リスト名a**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a[0] | a[1] | a[2] | a[3] | a[4] | a[5] | a[6] | a[7] | a[8] | a[9] |
| 56 | 3 | 62 | 17 | 87 | 22 | 36 | 83 | 21 | 12 |

**（１）次の計算をしてください。**

**①　a[3]+a[7]=**

**②　すべての要素を加えるといくつになりますか。**

**③　添え字が奇数のものだけ加えると合計でいくつになりますか。**

**（２）「3番目の要素と7番目の要素の足し算を計算する」プログラムと「リストのすべての要素の合計を計算する」プログラムを作成し，上の①と②の結果と一致しているか確認しよう。**

**プログラム①**

|  |  |
| --- | --- |
| 1  2  3  4 | a = [56,3,62,17,87,22,36,83,21,12] #リストaの定義  goukei = 0  goukei = a[3]+a[7]  print(goukei) |

**プログラム②**

|  |  |
| --- | --- |
| 1  2  3  4  5 | a = [56,3,62,17,87,22,36,83,21,12]  goukei = 0  for i in range(0,10,1):  goukei = goukei+a[i] #a[0]～a[9]まで全て加えていく  print(goukei) |

**（３）「リストの添え字が奇数のものだけ加える」プログラムを作成し，上の③の結果と一致しているか確認しよう。**

**リストを使ってデータをまとめて処理をしよう**

教師用

　　年　　組(　　)　　　　　　　　　.

**プログラムが大規模になると扱うデータの数も膨大になる。ここではデータを一つのかたまりとして処理をするためのリストについて学ぼう。**

【指導ポイント】

リストは0番目から数えることを強調して説明する。

**リスト名a**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a[0] | a[1] | a[2] | a[3] | a[4] | a[5] | a[6] | a[7] | a[8] | a[9] |
| 56 | 3 | 62 | 17 | 87 | 22 | 36 | 83 | 21 | 12 |

**（１）次の計算をしてください。**

**①　a[3]+a[7]=100**

**②　すべての要素を加えるといくつになりますか。　399**

**③　添え字が奇数のものだけ加えると合計でいくつになりますか。　137**

**（２）「3番目の要素と7番目の要素の足し算を計算する」プログラムと「リストのすべての要素の合計を計算する」プログラムを作成し，上の①と②の結果と一致しているか確認しよう。**

**プログラム①**

|  |  |
| --- | --- |
| 1  2  3  4 | a = [56,3,62,17,87,22,36,83,21,12] #リストaの定義  goukei = 0  goukei = a[3]+a[7]  print(goukei)  【指導ポイント】  0番目から順番に加えていく様子を丁寧に説明する。 |

**プログラム②**

|  |  |
| --- | --- |
| 1  2  3  4  5 | a = [56,3,62,17,87,22,36,83,21,12]  goukei = 0  for i in range(0,10,1):  goukei = goukei+a[i] #a[0]～a[9]まで全て加えていく  print(goukei) |

**（３）「リストの添え字が奇数のものだけ加える」プログラムを作成し，上の③の結果と一致しているか確認しよう。**

【評価ポイント】

プログラムが正しく作成されているか。

**a = [56,3,62,17,87,22,36,83,21,12]**

**goukei = 0**

【指導ポイント】

奇数番目のものだけが正しくピックアップされているか確認させる。

**for i in range(0,10,1):**

**if(i%2==1):**

**goukei = goukei+a[i]**

**print(goukei)**

***第3章学習14\_展開2ワークシート***

**乱数を使ったプログラムを作成しよう**

　　年　　組(　　)　　　　　　　　　.

**乱数を使うことによってあえてランダムに処理や結果が変化するプログラムを作成することが可能となります。ここでは乱数について学ぼう。**

**（１）実際に処理結果がランダムに表れる例を考えてみよう。**

|  |  |  |
| --- | --- | --- |
| 例 | ランダムな処理 | ランダムでないとどうなるか |
| (例)  くじ引き | くじの結果がランダム | いつも同じくじが引かれる |
|  |  |  |

**（２）「20%の確率で当たりと表示される」プログラムを作成しよう。プログラムが完成したら複数回実行し，当たりとはずれが表示されることを確認しよう。**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | import random #randomモジュールを読み込む  r = random.randrange(10) #0～9までの整数をランダムに発生  if r <= 1:  print("当たり")  else:  print("はずれ") |

**（３）先ほどのプログラムを改良し，「70%の確率で当たりと表示される」プログラムを作成しよう。また，乱数rの値は0から9までとし，rがいくつの時に当たりと表示するかを下の表にまとめよう。**

|  |  |  |  |
| --- | --- | --- | --- |
| rの値 | 当たり or はずれ | rの値 | 当たり or はずれ |
| 0 |  | 5 |  |
| 1 |  | 6 |  |
| 2 |  | 7 |  |
| 3 |  | 8 |  |
| 4 |  | 9 |  |

**乱数を使ったプログラムを作成しよう**

教師用

　　年　　組(　　)　　　　　　　　　.

**乱数を使うことによってあえてランダムに処理や結果が変化するプログラムを作成することが可能となります。ここでは乱数について学ぼう。**

**（１）実際に処理結果がランダムに表れる例を考えてみよう。**

|  |  |  |
| --- | --- | --- |
| 例 | ランダムな処理 | ランダムでないとどうなるか |
| (例)  くじ引き | くじの結果がランダム | いつも同じくじが引かれる  【指導ポイント】  生徒から考えが出なければ，教師からレジの例を挙げ，他の意見を促す。 |
| レジのシミュレーション | 客が来店するタイミングが  ランダム | いつも同じ時間に客が来る |

**（２）「20%の確率で当たりと表示される」プログラムを作成しよう。プログラムが完成したら複数回実行し，当たりとはずれが表示されることを確認しよう。**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | import random #randomモジュールを読み込む  r = random.randrange(10) #0～9までの整数をランダムに発生  if r <= 1:  【指導ポイント】  10個の乱数のなかから7個当たりとなるように指定すればよいことを事前に伝える。  print("当たり")  else:  print("はずれ") |

**（３）先ほどのプログラムを改良し，「70%の確率で当たりと表示される」プログラムを作成しよう。また，乱数rの値は0から9までとし，rがいくつの時に当たりと表示するかを下の表にまとめよう。**

|  |  |  |  |
| --- | --- | --- | --- |
| rの値 | 当たり or はずれ | rの値 | 当たり or はずれ |
| 0 | 当たり | 5 | 当たり |
| 1 | 当たり | 6 | 当たり |
| 2 | 当たり | 7 | はずれ |
| 3 | 当たり | 8 | はずれ |
| 4 | 当たり | 9 | はずれ |

**import random #randomモジュールを読み込む**

**r = random.randrange(10) #0～9までの整数をランダムに発生**

**if r <= 6:**

【評価ポイント】

当たりorはずれの列がｒの値に応じて正しく記載されているか。

**print("当たり")**

**else:**

**print("はずれ")**

***第3章学習14\_展開3ワークシート***

**複雑なプログラムを機能ごとに分割しよう**

　　年　　組(　　)　　　　　　　　　.

**実際に行われているシステム開発の場面のプログラムは大規模なものになるため，機能ごとに分割することが行われる。分割する理由とその方法について学ぼう。**

**（１）大規模なプログラムではなぜ機能ごとに分割することが望ましいのかその理由を考えてみよう。**

|  |  |
| --- | --- |
| 理由１ |  |
| 理由２ |  |

**（２）次のプログラムは「リスト内のすべての要素の合計を求める」プログラムである。このプログラムの動作を確認し，関数listgoukeiの引数と戻り値を下の表にまとめよう。**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | def listgoukei(a): #合計を求める関数listgoukeiを作成  goukei = 0  for i in range(0,len(a),1):  goukei = goukei+a[i]  return goukei  a = [56,3,62,17,87,22,36,83,21,12]  goukei = listgoukei(a) #作った関数listgoukeiを呼び出し  print(goukei) |

|  |  |  |
| --- | --- | --- |
| 関数の機能 | 関数の引数 | 関数の戻り値 |
| リスト内の要素の合計を  求める |  |  |

**（３）先ほどのプログラムを参考にし，「リスト内のすべての要素から最小値を求める」プログラムを作成しよう。ただし以下の機能について関数listminという名前で作成してください。**

|  |  |  |
| --- | --- | --- |
| 関数の機能 | 関数の引数 | 関数の戻り値 |
| リスト内の要素から最小値を求める |  |  |

**複雑なプログラムを機能ごとに分割しよう**

教師用

　　年　　組(　　)　　　　　　　　　.

**実際に行われているシステム開発の場面のプログラムは大規模なものになるため，機能ごとに分割することが行われる。分割する理由とその方法について学ぼう。**

【指導ポイント】

大規模なコードを作成するとどのような困った場面が出てくるか考えさせる。

**（１）大規模なプログラムではなぜ機能ごとに分割することが望ましいのかその理由を考えてみよう。**

|  |  |
| --- | --- |
| 理由１ | 機能ごとに分割することによりコードが短くなり，可読性が高まる。 |
| 理由２ | 機能ごとに分割することにより他人に再利用してもらいやすくなる。 |

**（２）次のプログラムは「リスト内のすべての要素の合計を求める」プログラムである。このプログラムの動作を確認し，関数listgoukeiの引数と戻り値を下の表にまとめよう。**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | def listgoukei(a): #合計を求める関数listgoukeiを作成  goukei = 0  【指導ポイント】  処理に必要なデータと処理後に得られるデータを強く意識させるようにする。  for i in range(0,len(a),1):  goukei = goukei+a[i]  return goukei  a = [56,3,62,17,87,22,36,83,21,12]  goukei = listgoukei(a) #作った関数listgoukeiを呼び出し  print(goukei) |

|  |  |  |
| --- | --- | --- |
| 関数の機能 | 関数の引数 | 関数の戻り値 |
| リスト内の要素の合計を  求める | 合計を求めたいリスト | リスト内の要素の合計 |

**（３）先ほどのプログラムを参考にし，「リスト内のすべての要素から最小値を求める」プログラムを作成しよう。ただし以下の機能について関数listminという名前で作成してください。**

|  |  |  |
| --- | --- | --- |
| 関数の機能 | 関数の引数 | 関数の戻り値 |
| リスト内の要素から最小値を求める | 最小値を求めたいリスト | リスト内の最小値の値 |

**def listmin(a):**

【評価ポイント】

関数の引数，関数の戻り値が適切に記載されている。

**min = a[0]**

**for i in range(1,len(a),1):**

**if(min>a[i]):**

**min = a[i]**

**return min**

**a = [56,3,62,17,87,22,36,83,21,12]**

**min = listmin(a)**

**print(min)**

***第3章学習15\_展開1ワークシート***

**線形探索について考えよう**

　　年　　組(　　)　　　　　　　　　.

**リストの中から値を探索する手法の一つである線形探索について学ぼう。**

**リスト名a**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| a[0] | a[1] | a[2] | a[3] | a[4] | a[5] | a[6] |
| 61 | 15 | 82 | 77 | 21 | 32 | 53 |

**（１）a[0]から順番に探索していく線形探索を行う場合，次の場合について考えよう。**

**①　77を探す場合は何回値を調べることとなりますか。**

**②　32を探す場合は何回値を調べることとなりますか。**

**③　100を探す場合は何回値を調べることとなりますか。**

**（２）次の線形探索のプログラムの動作を確認し，11行目のpの値を変更しながら上の①～③を確認しよう。**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | def linsearch(a,p):  c = 0  for i in range(0,len(a),1):  c = c+1  if a[i]==p:  print("見つかりました")  print(c,"回探索しました")  break  a = [61,15,82,77,21,32,53]  p = 77  linsearch(a,p) |

**線形探索について考えよう**

教師用

　　年　　組(　　)　　　　　　　　　.

**リストの中から値を探索する手法の一つである線形探索について学ぼう。**

**リスト名a**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| a[0] | a[1] | a[2] | a[3] | a[4] | a[5] | a[6] |
| 61 | 15 | 82 | 77 | 21 | 32 | 53 |

**（１）a[0]から順番に探索していく線形探索を行う場合，次の場合について考えよう。**

**①　77を探す場合は何回値を調べることとなりますか。**

４回　（a[0]から順番に探索していくので，77は4番目）

【指導ポイント】

前から順に調べるため，回数に差が付くことを強調する。

**②　32を探す場合は何回値を調べることとなりますか。**

６回　（a[0]から順番に探索していくので，32は6番目）

**③　100を探す場合は何回値を調べることとなりますか。**

７回　（100はリストにないので，探索はa[0]～a[6]までのすべての要素について行われる）

**（２）次の線形探索のプログラムの動作を確認し，11行目のpの値を変更しながら上の①～③を確認しよう。**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | def linsearch(a,p):  c = 0  for i in range(0,len(a),1):  c = c+1  if a[i]==p:  print("見つかりました")  print(c,"回探索しました")  break  a = [61,15,82,77,21,32,53]  p = 77  linsearch(a,p) |

【評価ポイント】

ｐを①から③で探索する値に設定して実行した際に画面に表示される回数の数字を書かせるなどして，実際にプログラムを動かして確認したことを評価する。

***第3章学習15\_展開2ワークシート***

**二分探索について考えよう**

　　年　　組(　　)　　　　　　　　　.

**リストの中から値を探索する手法としていくつかあるが，そのうちの一つである二分探索について学ぼう。**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| a[0] | a[1] | a[2] | a[3] | a[4] | a[5] | a[6] |
| 25 | 33 | 43 | 51 | 66 | 71 | 88 |

**リスト名a**

**（１）昇順に並べ替えられているデータに対して二分探索を行う場合，次の場合について考えよう。**

**①　33を探す場合は何回値を調べることとなりますか。**

**②　88を探す場合は何回値を調べることとなりますか。**

**③　100を探す場合は何回値を調べることとなりますか。**

**（２）次の二分探索のプログラムの動作を確認し，19行目のpの値を変更しながら上の①～③の確認をしよう。**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | def binsearch(a,p):  i = 0  j = len(a)-1  c = 0  while i<=j:  m = int((i+j)/2)  c = c+1  if a[m]==p:  print("見つかりました")  print(c,"回探索しました")  break  else:  if a[m]>p:  j=m-1  else:  i=m+1  a = [25,33,43,51,66,71,88]  p = 43  binsearch(a,p) |

**二分探索について考えよう**

教師用

　　年　　組(　　)　　　　　　　　　.

**リストの中から値を探索する手法としていくつかあるが，そのうちの一つである二分探索について学ぼう。**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| a[0] | a[1] | a[2] | a[3] | a[4] | a[5] | a[6] |
| 25 | 33 | 43 | 51 | 66 | 71 | 88 |

**リスト名a**

【指導ポイント】

線形探索と比較して，調べる回数に大きな差は起こりにくいことを強調する。

**（１）昇順に並べ替えられているデータに対して二分探索を行う場合，次の場合について考えよう。**

**①　33を探す場合は何回値を調べることとなりますか。**

２回　（a[0]とa[6]の中央はa[3]=51，33の方が小さいので，求める値は，a[0]からa[3]の間にある。その中央はa[1]=33，これは探索地と一致するので検索回数は2回。）

**②　88を探す場合は何回値を調べることとなりますか。**

３回　（a[0]とa[6]の中央はa[3]=51，これは88より小さいので，求める値は，a[4]からa[6]の間にある。その中央はa[5]=71，これも88より小さいので，求める値はa[6]に決定する。）

**③　100を探す場合は何回値を調べることとなりますか。**

３回　（②と同様に操作し，最後にa[6]は100と等しくないので，リストに100はないと判別できる。）

**（２）次の二分探索のプログラムの動作を確認し，19行目のpの値を変更しながら上の①～③の確認をしよう。**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | def binsearch(a,p):  i = 0  j = len(a)-1  c = 0  while i<=j:  m = int((i+j)/2)  c = c+1  if a[m]==p:  print("見つかりました")  print(c,"回探索しました")  break  else:  if a[m]>p:  j=m-1  【評価ポイント】   * 作成したプログラムが正しく実行できているか。 * 手で探索した回数と一致しているか。   else:  i=m+1  a = [25,33,43,51,66,71,88]  p = 43  binsearch(a,p) |

***第3章学習15\_展開3ワークシート***

**線形探索と二分探索の比較をしよう**

　　年　　組(　　)　　　　　　　　　.

**アルゴリズムはその場面ごとに有効なものが変わってくることがある。線形探索と二分探索の比較をしてみよう。**

**（１）線形探索と二分探索の最大探索回数を比較してみよう。**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| データ数 | 2 | 10 | 100 | 1000 | 10000 |
| 線形探索での最大探索回数 | 2 |  |  |  |  |
| 二分探索での最大探索回数 | 2 |  |  |  |  |

**（２）上の結果から探索するデータ数が多いほどどのようなことになるかまとめよう。**

|  |
| --- |
|  |

**（３）上のことから特にデータ数が多い場合は（　　　　　　）探索は有用な場合が多い。ただ常にこの探索方法が最善であるとは限らない。どのような場合に遅くなる可能性があるのか考えよう。**

|  |
| --- |
|  |

**線形探索と二分探索の比較をしよう**

教師用

　　年　　組(　　)　　　　　　　　　.

**アルゴリズムはその場面ごとに有効なものが変わってくることがある。線形探索と二分探索の比較をしてみよう。**

【評価ポイント】

最大探索回数が正しく求められているか。

**（１）線形探索と二分探索の最大探索回数を比較してみよう。**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| データ数 | 2 | 10 | 100 | 1000 | 10000 |
| 線形探索での最大探索回数 | 2 | 10 | 100 | 1000 | 10000 |
| 二分探索での最大探索回数 | 2 | 4 | 7 | 10 | 14 |

【指導ポイント】

二分探索では１回値を調べると、調べるデータの大きさが半分になっていくことを強調する。

**（２）上の結果から探索するデータ数が多いほどどのようなことになるかまとめよう。**

|  |
| --- |
| 二分探索の方がデータ数が多くなっても調べる回数が多くなりにくい。  従ってデータ数が多い場合は二分探索が有用である。 |

**（３）上のことから特にデータ数が多い場合は（　二分　　）探索は有用な場合が多い。ただ常にこの探索方法が最善であるとは限らない。どのような場合に遅くなる可能性があるのか考えよう。**

|  |
| --- |
| 二分探索を行うには事前にデータを並べ替えておく必要がある。  データを並べ替えるのに時間がかかる場合は，線形探索の方が有用な場合もある。  【評価ポイント】  ・データ数が大きくなることで有用なアルゴリズムが変わることが理解できているか。  ・全ての場合について有用なアルゴリズムはないことを，二分探索の例を通じて理解できたか。  【指導ポイント】  二分探索を行うためには並べ替えられている必要があることを説明し，常に二分探索が最善であるわけではないことを強調する。 |

***第3章学習16\_展開1\_ワークシート***

**サイコロのシミュレーションプログラムを実行してみよう**

　　年　　組(　　)　　　　　　　　　.

**（１）次のプログラムはサイコロのシミュレーションを行うプログラムである。空欄を埋めて完成させ，実行してみよう。**

|  |  |
| --- | --- |
| 行番号 | コード(python) |
| 1  2  3  4  5  6  7  8  9 | import numpy as np #整数をカウントするための関数呼び出し  import numpy.random as rd #乱数を発生させる関数の呼び出し  saikoro = rd.randint(1, 6+1, 100) #サイコロを100回振る  print(saikoro) #乱数の値を表示  deme = [ ] #出目の数を数える配列  for i in range(6):  deme.append(np.count\_nonzero(saikoro== i+1 )) #出現回数を数えて配列に追加  print("出現数:",deme) #乱数の値を表示 |

**（２）サイコロを振る回数を変えてみて，結果を比べてみよう。**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ｻｲｺﾛ  の目 | サイコロの目の出現カウント | | 出現確率（％） | |
| １００回の時 | ５０００回の時 | １００回の時 | ５０００回の時 |
| １ | 回出現 | 回出現 | ％ | ％ |
| ２ | 回出現 | 回出現 | ％ | ％ |
| ３ | 回出現 | 回出現 | ％ | ％ |
| ４ | 回出現 | 回出現 | ％ | ％ |
| ５ | 回出現 | 回出現 | ％ | ％ |
| ６ | 回出現 | 回出現 | ％ | ％ |

**（３）プログラムの実行結果を考えてみよう。**

**・回数が少ない時と多い時，出現確率はどうなるか？**

**・手でサイコロを振った時と比べて，良い点や悪い点はあるか？**

**サイコロのシミュレーションプログラムを実行してみよう**

教師用

　　年　　組(　　)　　　　　　　　　.

**（１）次のプログラムはサイコロのシミュレーションを行うプログラムである。空欄を埋めて完成させ，実行してみよう。**

|  |  |
| --- | --- |
| 行番号 | コード(python) |
| 1  2  3  4  5  6  7  8  9 | import numpy as np #整数をカウントするための関数呼び出し  import numpy.random as rd #乱数を発生させる関数の呼び出し  saikoro = rd.randint(1, 6+1, 100) #サイコロを100回振る  print(saikoro) #乱数の値を表示  deme = [ ] #出目の数を数える配列  for i in range(6):  deme.append(np.count\_nonzero(saikoro==i+1)) #出現回数を数えて配列に追加  【指導ポイント】  すでにリストsaikoroに乱数が全て格納されている事やループ用変数iに着目させる。  print("出現数:",deme) #乱数の値を表示 |

**（２）サイコロを振る回数を変えてみて，結果を比べてみよう。**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ｻｲｺﾛ  の目 | サイコロの目の出現カウント | | 出現確率（％） | |
| １００回の時 | ５０００回の時 | １００回の時 | ５０００回の時 |
| １ | 回出現 | 回出現 | ％ | ％ |
| ２ | 回出現 | 回出現 | ％ | ％ |
| ３ | 回出現 | 回出現 | ％ | ％ |
| ４ | 回出現 | 回出現 | ％ | ％ |
| ５ | 回出現 | 回出現 | ％ | ％ |
| ６ | 回出現 | 回出現 | ％ | ％ |

**（３）プログラムの実行結果を考えてみよう。**

**・回数が少ない時と多い時，出現確率はどうなるか？**

出現確率は１～６がほぼ均等になることを確認する。

**・手でサイコロを振った時と比べて，良い点や悪い点はあるか？**

プログラミングによって手間がかかる作業が軽減されている事，コンピュータによって繰り返し試行を膨大な回数実行可能な事を確認する。

***第3章学習16\_展開2\_ワークシート***

**乱数のシミュレーションをしてみよう**

　　年　　組(　　)　　　　　　　　　.

**（１）班に分かれて，サイコロはどのようなことに使えるか考えてみよう。**

|  |  |  |
| --- | --- | --- |
|  | 何に使える？ | サイコロをどのように使う？ |
| アイデア |  |  |

**（２）他の班の意見も聞いてメモしよう。**

|  |  |  |
| --- | --- | --- |
|  | 何に使える？ | サイコロをどのように使う？ |
| アイデア |  |  |

**（３）実際にサイコロを振って出た目を数えてみて，班ごとに集計しよう。**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ｻｲｺﾛ  の目 | サイコロの目の出現カウント | | 出現確率（％） | |
| 班別 | 全体 | 班別 | 全体 |
| １ | 回出現 | 回出現 | ％ | ％ |
| ２ | 回出現 | 回出現 | ％ | ％ |
| ３ | 回出現 | 回出現 | ％ | ％ |
| ４ | 回出現 | 回出現 | ％ | ％ |
| ５ | 回出現 | 回出現 | ％ | ％ |
| ６ | 回出現 | 回出現 | ％ | ％ |

**（４）班ごとの結果と全体を足した結果で出現確率はどうなるか？**

**サイコロ以外で乱数のシミュレーションに使える現象をあげてみよう。**

**乱数のシミュレーションをしてみよう**

教師用

　　年　　組(　　)　　　　　　　　　.

**（１）班に分かれて，サイコロはどのようなことに使えるか考えてみよう。**

|  |  |  |
| --- | --- | --- |
|  | 何に使える？ | サイコロをどのように使う？ |
| アイデア | どちらか１方に決める。  じゃんけんの代わりに使う。  データの抽出につかう。 | サイコロの偶数か奇数かで決める。  出た目の大きい方が勝ち。  何回か振って足した値で抽出するデータを決める。 |

**（２）他の班の意見も聞いてメモしよう。**

【指導ポイント】

サイコロの目をどのように活用するのか，サイコロを何度振って使うのかなどを助言する。

|  |  |  |
| --- | --- | --- |
|  | 何に使える？ | サイコロをどのように使う？ |
| アイデア |  |  |

【指導ポイント】

「サイコロは公平に使える道具なのだろうか？結果に偏りはあるのだろうか？」と生徒に問いかける。

**（３）実際にサイコロを振って出た目を数えてみて，班ごとに集計しよう。**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ｻｲｺﾛ  の目 | サイコロの目の出現カウント | | 出現確率（％） | |
| 班別 | 全体 | 班別 | 全体 |
| １ | 回出現 | 回出現 | ％ | ％ |
| ２ | 回出現 | 回出現 | ％ | ％ |
| ３ | 回出現 | 回出現 | ％ | ％ |
| ４ | 回出現 | 回出現 | ％ | ％ |
| ５ | 回出現 | 回出現 | ％ | ％ |
| ６ | 回出現 | 回出現 | ％ | ％ |

**（４）班ごとの結果と全体を足した結果で出現確率はどうなるか？**

**サイコロ以外で乱数のシミュレーションに使える現象をあげてみよう。**

【評価ポイント】

サイコロが一様乱数の道具として機能しなければならない事を理解できているか。（理解の進んでいる生徒には，コイン投げなども同じような使い方をしている点などを助言する。）

***第3章学習17\_展開1\_ワークシート***

**放物運動の数式モデルを考えよう**

　　年　　組(　　)　　　　　　　　　.

**空気抵抗を考慮しない放物運動における物体の位置のシミュレーションはどのように行えるだろうか。角度θで投げ上げられた物体の現在位置と速度を考えて次の瞬間の（微小時間後）位置を求める数式モデルを作成してみよう。**

**（１）現在時間と微小時間後の差を時間間隔と呼ぶ。**

**水平方向（ｘ方向）の微小時間後の位置について，空欄を埋めて式を完成させよう。**

※ 現在の速度と，次の時点では速度の変化も考えらる。シミュレーションでは，現時点から次の時点を計算するため，その間の速度は平均速度と呼ぶ。速度に変化が無い場合は速度＝平均速度であり，変化がある場合には（現時点の速度＋微小時間後の速度）／２と考えられる。（※例えば，速度＝２から速度＝１に減速する間の平均速度は１．５となる。）

※ 水平方向と鉛直方向の初速度は，角度θを三角関数で分解する。

水平方向（ｘ方向）の速度＝初速度×cos(θ)　　　鉛直方向（y方向）の速度＝初速度×sin（θ）

|  |  |
| --- | --- |
|  | 数式モデル |
| 水平方向  の微小時間後の位置 | 平均速度　＝　水平方向の初速度  微小時間後の位置　＝　現在の水平位置　＋　平均速度　×　時間間隔 |

**（２）垂直方向（ｙ方向）について空欄を埋めて式を完成させよう。**

|  |  |
| --- | --- |
|  | 数式モデル |
| 垂直方向  の微小時間後の位置 | 鉛直方向には，重力加速度がかかり，速度が下がっていきます。  微小時間後の速度　＝　現在の速度　－　重力加速度（G）　×　時間間隔  平均速度　＝　（現在の速度　＋　微小時間後の速度）　／　２  微小時間後の位置　＝　現在の位置　＋　平均速度　×　時間間隔 |

**放物運動の数式モデルを考えよう**

教師用

　　年　　組(　　)　　　　　　　　　.

**空気抵抗を考慮しない放物運動における物体の位置のシミュレーションはどのように行えるだろうか。角度θで投げ上げられた物体の現在位置と速度を考えて次の瞬間の（微小時間後）位置を求める数式モデルを作成してみよう。**

**（１）現在時間と微小時間後の差を時間間隔と呼ぶ。**

**水平方向（ｘ方向）の微小時間後の位置について，空欄を埋めて式を完成させよう。**

※ 現在の速度と，次の時点では速度の変化も考えらる。シミュレーションでは，現時点から次の時点を計算するため，その間の速度は平均速度と呼ぶ。速度に変化が無い場合は速度＝平均速度であり，変化がある場合には（現時点の速度＋微小時間後の速度）／２と考えられる。（※例えば，速度＝２から速度＝１に減速する間の平均速度は１．５となる。）

※ 水平方向と鉛直方向の初速度は，角度θを三角関数で分解する。

水平方向（ｘ方向）の速度＝初速度×cos(θ)　　　鉛直方向（y方向）の速度＝初速度×sin（θ）

|  |  |
| --- | --- |
|  | 数式モデル |
| 水平方向  の微小時間後の位置 | 平均速度　＝　水平方向の初速度  微小時間後の位置　＝　現在の水平位置　＋　平均速度　×　時間間隔 |

**（２）垂直方向（ｙ方向）について空欄を埋めて式を完成させよう。**

|  |  |
| --- | --- |
|  | 数式モデル |
| 垂直方向  の微小時間後の位置 | 鉛直方向には，重力加速度がかかり，速度が下がっていきます。  微小時間後の速度　＝　現在の速度　－　重力加速度（G）　×　時間間隔  平均速度　＝　（現在の速度　＋　微小時間後の速度）　／　２  微小時間後の位置　＝　現在の鉛直位置　＋　平均速度　×時間間隔 |

【指導ポイント】

物理現象のシミュレーションであるが，式の一つ一つを丁寧に説明すれば難しくない。

目的はシミュレーションによって物体の動きを予測する事であることを明確にしておく。

***第3章学習17\_展開2,3\_ワークシート***

**放物運動のシミュレーションプログラムを作ろう**

　　年　　組(　　)　　　　　　　　　.

**（１）数式モデルを作成できれば，プログラムにそのモデルを当てはめることができる。空欄を埋めてプログラムを完成させ，実行してみよう。**

※プログラミングでは，角度を使うときにラジアン（弧度法）を使う。

１ラジアンはπ／180°であり，例えば４５°をラジアンにすると４５×π／１８０°で計算することとなる。

|  |  |
| --- | --- |
|  | プログラムコード（python） |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | import math as math 　#数値計算ライブラリ  import matplotlib.pyplot as plt 　#グラフ描画ライブラリ  dt = 0.01 　#微小時間(時間間隔)  v0 = 30 　#初速度  g = 9.8 　#重力加速度  x = [0] 　#水平位置の初期値は0  y = [0]　 #鉛直位置の初期値は0  angle = 45.0 \* math.pi / 180.0　 #投げ上げ角度  vx = [v0\*math.cos(angle)]　 #水平方向の初速度  vy = [v0\*math.sin(angle)] 　#鉛直方向の初速度  for i in range(1000):  vx.append(vx[i]) 　#微小時間後の水平方向の速度  vy.append(vy[i]- g\*dt ) 　#微小時間後の鉛直方向の速度  x.append(x[i]+ vx[i]\*dt )　 #微小時間後の水平位置  y.append(y[i]+(vy[i]+vy[i+1])/2.0\*dt)　 #微小時間後の鉛直位置  if y[i] < 0 :　 #もし鉛直位置が0を下回ったら  break 　#ループ中断  plt.plot(x,y)　 #位置の配列をプロット  plt.title("parabollic motion")　 #グラフのタイトル  plt.xlabel("distance")　 #x軸ラベル  plt.ylabel("height")　 #y軸ラベル  plt.show() |

**（２）このプログラムでは，鉛直方向の位置が０を下回ったら計算を中断して終わるようになっている。プログラムを実行した結果を確認できたら，投げ上げ角度を変更しながら実行し，どの角度が最も遠くに飛ばせるのか，どの角度が最も高く飛ばせるのかを確認してみよう。**

|  |  |
| --- | --- |
| 最も遠くに飛んだ角度 | 最も高く飛んだ角度 |
|  |  |

**放物運動のシミュレーションプログラムを作ろう**

教師用

　　年　　組(　　)　　　　　　　　　.

**（１）数式モデルを作成できれば，プログラムにそのモデルを当てはめることができる。空欄を埋めてプログラムを完成させ，実行してみよう。**

※プログラミングでは，角度を使うときにラジアン（弧度法）を使う。

１ラジアンはπ／180°であり，例えば４５°をラジアンにすると４５×π／１８０°で計算することとなる。

【指導ポイント】

プログラムを動かすことだけに集中させず，常に数式モデルとの対応を確認させながら，コードを理解させる。

|  |  |
| --- | --- |
|  | プログラムコード（python） |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | import math as math # 数値計算ライブラリ  import matplotlib.pyplot as plt # グラフ描画ライブラリ  dt = 0.01 # 微小時間( 時間間隔)  v0 = 30 # 初速度  g = 9.8 # 重力加速度  x = [0] # 水平位置の初期値0  y = [0] # 鉛直位置の初期値は0  angle = 45.0 \* math.pi / 180.0 # 投げ上げ角度  vx = [v0\*math.cos(angle)] # 水平方向の初速度  vy = [v0\*math.sin(angle)] # 鉛直方向の初速度  for i in range(1000):  vx.append(vx[i]) # 微小時間後の水平方向の速度  vy.append(vy[i]-g\*dt) # 微小時間後の鉛直方向の速度  x.append(x[i]+vx[i]\*dt) # 微小時間後の水平位置  y.append(y[i]+(vy[i]+vy[i+1])/2.0\*dt) # 微小時間後の鉛直位置  if y[i] < 0 : # もし鉛直位置が0 を下回ったら  break # ループ中断  plt.plot(x,y) # 位置の配列をプロット  plt.title("parabollic motion") # グラフのタイトル  plt.xlabel("distance") #x 軸ラベル  plt.ylabel("height") #y 軸ラベル  plt.show() |

**（２）このプログラムでは，鉛直方向の位置が０を下回ったら計算を中断して終わるようになっている。プログラムを実行した結果を確認できたら，投げ上げ角度を変更しながら実行し，どの角度が最も遠くに飛ばせるのか，どの角度が最も高く飛ばせるのかを確認してみよう。**

|  |  |
| --- | --- |
| 最も遠くに飛んだ角度 | 最も高く飛んだ角度 |
| ４５度 | ０度 |

【評価ポイント】

数式モデルとプログラムとの対応付けの理解ができているか。

※理解が進まない生徒に対しては，学習内容の振り返りや変数表の作成などを促すことで，理解を深められるように配慮する。